HELSINKI UNIVERSITY OF TECHNOLOGY
Department of Electrical and Communications Engineering
Laboratory of Space Technology

Pekka Ahtonen

Flood Detection by Spaceborne Radar

Master's Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology.

Espoo, 10th May 2005

Supervisor: Professor Martti Hallikainen
Instructor: Professor Jouni Pulliainen

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Pekka Ahtonen
Name of the Thesis:	Flood Detection by Spaceborne Radar
Date:	10th May 2005 **Number of pages:** 66 + 4
Department:	Electrical and Communications Engineering
Professorship:	S-92 Space Technology
Supervisor:	Prof. Martti Hallikainen
Instructor:	Prof. Jouni Pulliainen

In this study an automatic flood detection system for spaceborne SAR images is proposed. The system is based on the Active Contour Model that is a statistical snake algorithm, utilising SAR image statistics in the delineation process. It has been shown that this method can struggle the noise and the wind induced effects in SAR images unlike the traditional thresholding method.

The robustness of the flood detection system is tested with SAR images containing inundated areas with rough surfaces caused by emergent vegetation, for example. Finally, the delineation accuracy is validated by using a high resolution airborne image as a reference. The tests suggest that on roughened inundated areas the detector outperforms traditional thresholding, though in these cases completely automatic processing cannot be achieved. In the validation test the delineation accuracy was found to be within 50 m with 95 % probability.

Keywords: flood detection, active contour

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Pekka Ahtonen
Työn nimi:	Flood Detection by Spaceborne Radar
Päivämäärä:	10. toukokuuta 2005 **Sivuja:** 66 + 4
Osasto:	Sähkö- ja tietoliikennetekniikan osasto
Professuuri:	S-92 Avaruustekniikka
Työn valvoja:	Prof. Martti Hallikainen
Työn ohjaaja:	Prof. Jouni Pulliainen

Tässä diplomityössä kehitettiin automaattinen järjestelmä tulvien havaitsemiseen synteettistä apertuuria käyttävien satelliittitutkien kuvista, ts. SAR-kuvista. Järjestelmä perustuu olemassaolevaan Active Contour -malliin, joka on tilastollinen snake-algoritmi. Malli on eristyisesti suunniteltu hyödyntämään SAR-kuvan pikselien jakauman ominaisuuksia alueetten erottamisessa. Aikaisemmissa tutkimuksissa on voitu osoittaa että tämä menetelmä pystyy kompensoimaan kohinan ja tuulen aiheuttamat ongelmat SAR kuvissa paremmin kuin esimerkiksi tavanomainen kynnystysmenetelmä.

Kehitetyn järjestelmän häiriönsietoa testattiin SAR kuvilla, jotka sisälsivät esim. kasvillisuuden ja peltojen karhentamia tulva-alueita. Lopuksi järjestelmän alueitten rajaustarkkuus validoitiin käyttämällä referenssinä korkearesoluutioista, lentokoneesta otettua kuvaa. Testien perusteellä kehitetty järjestelmä pystyi karheapintaisilla tulva-alueilla parempaan rajaustulokseen kuin kynnystysmenetelmä. Tarkkuusvalidoinnissa rajaustuloksen todettiin olevan 50 m sisällä referenssistä 95 % todennäköisyydellä.

Avainsanat: tulvien havainnointi, active contour -algoritmi

Acknowledgements

I would like to express my gratitude to all who have given a hand and motivated me to push through this "interesting and very educational mission". In the first place special thanks go to Jenni and to my family who have provided much of the force needed for this project. Secondly I'd like to express my thanks to Stian Solbø who offered a local insight to the places in Tromsø during my one week work trip in NORUT. Por último les quiero dar las gracias a ellas en Sudamérica que me han animado a completar este grado.

Official thanks go to Mikko Sane in Finnish Environmental Institute as he provided the documentation and the ground truth data of the historical flood events in Finland.

Otaniemi, 10th May 2005

Pekka Ahtonen

Table of Contents

List of Symbols

A	electromagnetic wave's amplitude
c	speed of light constant
D_a	antenna's physical length
I	electromagnetic wave's intensity
k	wave number
$N(\mu, \sigma^2)$	normal distribution with mean μ and variance σ^2
$O(.)$	"Big-Oh" notation for both time and memory complexity
R_E	the Earth's mean radius
v	variance
v'	sample variance
$\Delta\phi$	phase difference
λ	wavelength
μ	mean
μ'	sample mean
ρ_x	azimuth resolution (along-track resolution)
ρ_y	range resolution (across-track resolution)
σ^0	differential backscattering coefficient or briefly backscattering coefficient
ϕ	electromagnetic wave's phase

List of Acronyms

ACIA	Arctic Climate Impact Assessment
AISA	Airborne Imaging Spectrometer for Applications
API	Application Interface
BDT	Binary Decision Tree
CCW	Counter-Clockwise
CW	Clockwise
DEM	Digital Elevation Model
EO	Earth Observation
EM	Electromagnetic
EU	European Union
FOV	Field Of View
GIS	Geoinformation System
GPS	Global Positioning System
GVF	Gradient Vector Flow
I/O	Input / Output
IDL	Interactive Data Language
JPL	Jet Propulsion Laboratory
LMS	Least Mean Squares
MAP	Maximum A Posteriori

MAV	Moving Average
ML	Maximum Likelyhood
MLE	Maximum Likelyhood Estimate
NASA	National Aeronautics and Space Administration
NIR	Near Infrared
RCS	Radar Cross Section
RGB	Red Green Blue
RMS	Root-Mean-Square
SAR	Synthetic Aperture Radar
SIR	Shuttle Imaging Radar
SLAR	Side Looking Airborne Radar
SLR	Side Looking Radar
SW	Software
TIR	Thermal Infrared
UML	Unified Modelling Language
UTM-35	Universal Transverse Mercator projection with centre longitude of 35 degrees
WGS-84	World Geodetic System 1984

Chapter 1

Introduction

Natural disasters like landslides, earthquakes and floods cause every year both economical and human losses. The global warming due to the burning of fossil fuels causes glacial melting, rises sea level, increases the variation in precipitation and thus affects to the global climate profoundly. Aforementioned effects were included among the ten key findings of the *ACIA report* [1] published on 8th November 2004 by the *Arctic Council*. Nations taking part to this intergovernmental forum are Canada, Denmark/Greenland/Faroe Islands, Finland, Iceland, Norway, Russia, Sweden and the United States. Additionally the Arctic Council comprises peoples' organisations, external observers and other international bodies.

The report itself focuses to the climate changes in the calotte area above 50 degrees latitude north, which corresponds to roughly the latitude of London, for example. Thus, the study area comprises Northern Europe, Siberia, Alaska and Canada's northern parts limiting to the level of the southernmost point of Hudson Bay. Some remarks from the complimentary list in the ACIA report: peoples living on the coastal regions will face increasing exposure to storms and floods, floods in mainland areas will manifest themselves in a bigger scale because of the increased precipitation, and also some other natural hazards are prone to increase in frequency and severity. Being not able to counteract the large scale natural hazards, its better to try minimise the losses by actively monitoring the environment and then acting before it is too late. For example, in case of floods, by forecasting a possible flood couple days in advance the authorities will have enough time to evacuate people from the risk areas.

Forecasts of discharges and water levels are usually derived from hydraulic and hydrological models for a certain watershed. Running a forecast model usually requires several input parameters including e.g. surface areas of water bodies on the upstream (when we are talking about forecasting a water level for a river). From the ground level a water body area could be determined, for example, by measuring enough GPS-points near the

waterline. In that way the area can be represented by a polygon whose size approximates the true area of the water body. However, having large water bodies the method is not convenient due to the required time and manpower for the operation.

From the air a water body extent can be determined by using aerial photos, optical or radar satellite images with a sufficient resolution. The obvious drawback with the first two alternatives is the vulnerability for the weather conditions. Having a lot of moisture in the air or even just light clouds can make it impossible to extract the water body extent properly from the image. Not to mention what is the situation with the weather conditions during the floods that are caused by heavy rains or storms.

Unlike the satellites carrying optical instruments SAR satellites are practically immune to the bad weather. In addition, global[1] coverage with high temporal and spatial frequency make them a feasible choice to monitor nature resources in many cases. In the SAR image it is possible to delineate water bodies' extents and thus monitor a certain area regularly providing input for the flood forecast models. However, due to the nature of the radar imaging process the interpretation of the SAR image is not very straightforward in every case. Mainly depending on polarisation, frequency and incidence angle the water areas in the SAR image may manifest themselves in various ways. For example, flooded agricultural fields with crops can cause significant increase in backscatter (at C-band) due to the double-reflection from the vegetation. In the SAR image this increase means brighter spots on normally nearly black water plains. Thus, the phenomena disturbs essentially the separation of water and land. The double-reflection from vegetation along with the other disturbing factors is discussed in detail in Sections 2.1 and 2.2.

The most flood-prone period in Finland spans from April to June. During that time floods can occur anywhere in Finland due to the melting of snow and the precipitation. Especially during early spring when the ground is still partly frozen both the meltwater and the precipitation increase the discharges in the rivers and thus raise the water levels as well. Even without the precipitation the meltwater is enough to raise the water level because the frozen ground cannot absorb it sufficiently. On the contrary, during summer lakes and swamp areas act as reservoirs for the precipitation so that the floods due to heavy raining are usually possible only in urban areas.

Naturally, the damages caused by a flood depend mainly on the landform and on the terrain of the area that is to be inundated. Especially vulnerable for floods is Pohjanmaa region, in Western Finland, whose terrain is characterised by vast flat plains. There even moderate floods can easily damage agricultural fields, roads, buildings and other infrastructure. Another flood-prone region experiencing yearly floods is the one following

[1]Because of the orbit configurations of operative SAR satellites, the coverage spans approximately from -80 to 80 degrees of latitude. Hence, only the polar regions cannot be imaged.

River Kemijoki in Lapland in Northern Finland. The vast swamp plains extending to the river bank make it easy for a flood to cover large land areas like in Pohjanmaa region. A spring flood in Western or Northern Finland can occur also as a consequence of an ice dam in a river. An ice dam blocking effectively the stream of water in a river can raise the water level rapidly and cause significant damage to the infrastructure nearby the river. In extreme cases such a dam can be exploded to restore the normal stream. Dams can also be caused by the frazil ice formating to a riverbed. These hanging ice dams can raise the water level in a river and build up a flood. The floods due to the hanging ice dams typically occur in Western or Southern Finland during autumn. Smaller floods can occur from spring to autumn due to the heavy raining under the stormy weather conditions. In cities heavy rains can suffocate the sewer system letting the rain water to flood to underground floors and cellars causing significant damages. Additional harms include inundated roads and pedestrian lanes. [2]

The focus in this work was to analyse the historical flood cases in Finland and use them in the development of a robust water area delineation system that uses SAR images as input. With the term "robust" it is meant that the system should be able to tolerate the disturbances due to the double-reflection or due to the wind in flooded areas. The historical floods that cover an area large enough to be detected in the SAR image have occurred mainly in the rural areas or in the backwoods in Western or in Northern Finland. The prominent feature of these flood plains is that they all are more or less disturbed by the double-reflection effect by the vegetation or by the partly inundated ploughed areas. On the agricultural fields the vegetation is mainly consisted of different crops. On the swamps in the backwoods the vegetation typically ranges from grass to twigs and bushes. In addition to the double-reflection effect, some historical flood cases exhibit wind induced ripple that manifests itself as directionally textured bright areas on the flood plains in the SAR image. In some cases the wind ripple extends over the whole flood plain making it very difficult to recognise the transition from land to water, even when carefully inspecting the image visually.

The set up for the development of the flood detection system was to search for an algorithm capable to tolerate pre-described disturbing factors to a reasonable extent. More exactly, a slight wind ripple or a minor double-reflection should not cause problems to the system but in extreme cases, like the wind ripple heavily disturbing the flood plain, the detection of water would not be guaranteed. The goals were to select a suitable algorithm, to develop the software for the processing of a geocoded scene to a vector water mask and finally to validate the spatial accuracy of the water mask by using ground truth data. A special focus was given to automatise the system as far as possible.

This Master's Thesis was written within the *Floodman* project (EU Contract ID: EVG1-

CT-2002-00085) that is supported by the *European Commission* under the Frame Project 5: Energy, Environment and Sustainable Development. Floodman's objective is to "Develop, demonstrate and validate an information system for cost effective flood forecasting and management by using EO data, in particular spaceborne SAR data, hydrological and hydraulic models and in-situ data". For information, look at Floodman's Web pages at `http://projects.itek.norut.no/floodman`.

Chapter 2

Theoretical background

2.1. Physical background

In the following two sections the principles of the radar imaging system, the SAR processing and the SAR image properties are discussed. More information can be found in [3, 4, 5, 6, 7], for example.

2.1.1. Synthetic Aperture Radar image

In this section the focus is in *imaging, monostatic, side-looking, pulse microwave* radars onboard a satellite or an aircraft. The term *monostatic* implies that the transmitting and the receiving antenna of a radar are aligned. In other words they are the same antenna. Beginning from the basics of the conventional *real aperture radar*, the focus is then turned to the principles of the *synthetic aperture radar* and the radar image pre-processing.

The *side-looking airborne radar* (SLAR) or the *side-looking radar* (SLR) has an imaging configuration in which the radar is looking to the *range direction* that is a direction both normal to the flight path and tilted towards the ground. The angle between the local vertical and the antenna is called the *incidence angle* or the *look angle*[1]. Figure 2.1 represents an illustration of the side-looking configuration of a spaceborne radar.

Depending on the *beam width* of the antenna and the incidence angle, a radar has a certain *footprint* or an *illuminated area* on the ground. For the microwave radar is an active instrument, it sends and receives electromagnetic pulses to form an image. Having transmitted a short pulse it switches itself to a receive mode. Now the beginning of the sent pulse reflecting back from the ground is coming from a point that is closest to the platform

[1]To be exact, *look angle* implies the direction to which the radar antenna is looking and *incidence angle* the angle in which the radar pulse hits the ground. These two angles may differ in case of spaceborne radars because of the Earth's curvature. However, in the following we use *incidence angle* for the both meanings.

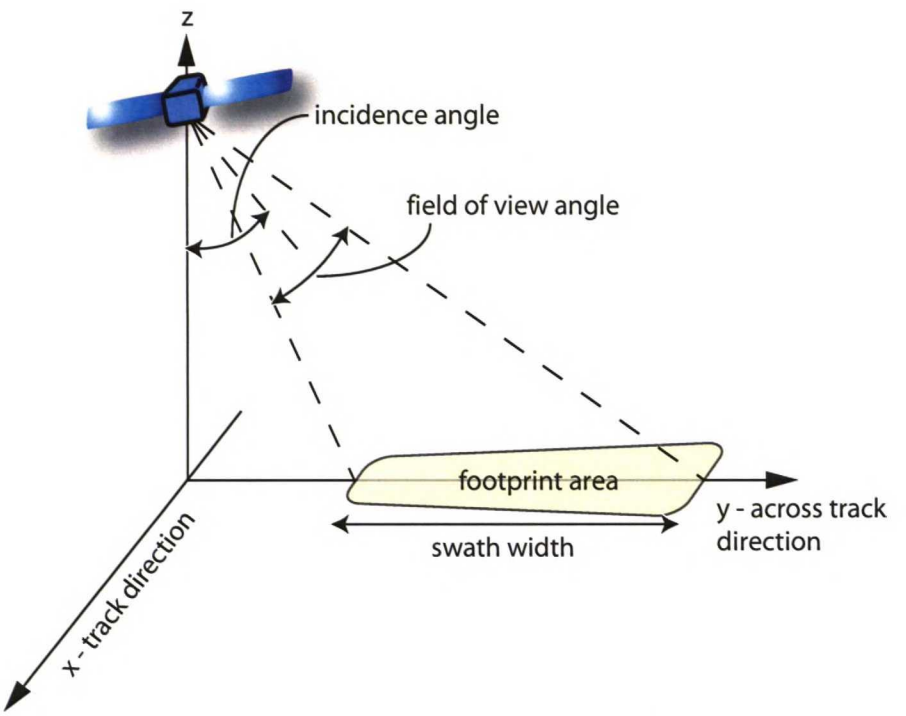

Figure 2.1: Typical imaging configuration of the spaceborne radar.

and vice versa. The amplitude of the pulse is subsequently decoded to the intensity values across the *swath width*, which is the width of the imaged area. Additionally, when the platform is moving with a constant speed, repeating the described process results the image strip that is built-up line by line. The conventional way to express the physical quantity measured by the radar is to use the *radar equation*

$$P_r = \frac{\lambda^2}{(4\pi)^3} \int_A \frac{P_t G^2 \sigma^0}{R^4} dA, \tag{2.1}$$

where λ is the wavelength, P_t and P_r are the respective sent and received power, G is the antenna gain, R is the *range* that is the distance between the radar and the target, σ^0 is the *differential backscattering coefficient* (or just the *backscattering coefficient* as the focus here is in the monostatic radar) and A is the whole measurement area. The radar equation can be derived briefly by starting with the equation for the received average power from many discrete scatterers

$$\overline{P}_r = \frac{\lambda^2}{(4\pi)^3} \sum_{i=1}^{N} \frac{P_{ti} G_i^2 \sigma_i}{R_i^4}, \tag{2.2}$$

where P_{ti}, G_i and R_i have the same meanings than in Equation 2.1, and σ_i is the *radar*

cross-section (RCS) that includes e.g. the gain and absorption properties of each discrete scatterer. Now we can define the backscattering coefficient as

$$\sigma^0 = \left\langle \frac{\sigma_i}{\Delta A_i} \right\rangle, \tag{2.3}$$

where ΔA_i is an area that is sufficiently small so that P_t, G and R are nearly constant over it. Because of the averaging, we can replace σ_i in Equation 2.2 by $\sigma_0 \Delta A_i$, perform limiting $\Delta A \rightarrow 0$ for the summation and finally take σ_0 out resulting

$$\sigma^0 = \frac{(4\pi)^3}{\lambda^2} \left(\frac{P_r}{P_t} \right) \frac{1}{\int_A G^2 / R^4 dA}. \tag{2.4}$$

For Equations 2.1 and 2.4 to be valid there must exist many individual scatterers in the area ΔA and even more in the area A over which the integral is taken. Assuming an antenna with a narrow beam width, the area A can be set to correspond to the one-pixel illuminated area in a non-smoothed image. Finally, the physical quantity that corresponds to a pixel's intensity value in the result image is σ^0 that is in effect the ratio of received and transmitted power per unit area, which is often expressed in decibels (dB). The *range resolution* in the real aperture radar image corresponds to the separation of two adjacent pixels on the ground in the range direction and is given by

$$\rho_y = \frac{c}{2B \sin \Theta}, \tag{2.5}$$

where B is the bandwidth of the radar pulse and Θ is the angle from the local vertical. Thus, the range resolution is best in the point that is nearest to the radar and worst in the point that is furthest from it. The *azimuth resolution* in the real aperture radar image corresponds to the separation of two points in the flight direction and is given by

$$\rho_x = \frac{h \Theta_x}{\cos \Theta}, \tag{2.6}$$

where h is the altitude of the radar, Θ_x is the antenna's effective beam width in the flight direction and Θ has the same meaning than in the previous equation. Thus, the azimuth resolution degrades as the range increases. The limiting factor for the azimuth resolution is the length of an antenna, D_a. By using relations $\Theta_x = \lambda / D_a$ and $r = h / \cos \Theta$, Equation 2.6 can be re-written to

$$\rho_x = \frac{\lambda r}{D_a}. \tag{2.7}$$

Consequently, to obtain better azimuth resolution the length of an antenna has to be increased.

The variation of the range inside each footprint causes that the data received by a radar have non-square resolution cells. Hence, a re-sampling or an interpolation pre-processing step is needed in order to store the data as a bitmap image. The other preprocessing steps will be covered later in this section, for now the focus is turned to the *synthetic aperture radar.*

In contrast to the *real aperture radar*, that was explained above, the *synthetic aperture radar* does not directly use the footprint of its antenna to produce image data. Instead, the whole image strip is formed at once, by using all recorded *complex number* data to computationally synthesise an aperture that covers the whole measured area. The complex number data includes both the *phase,* which is not stored by the conventional radar, and the *amplitude* of each backscattered electromagnetic wave.

The SAR image forming can be divided into a range and an azimuth processing directions. The *range processing* uses time delay data and the *azimuth processing* frequency and phase difference data to depict the position where a certain pulse originated. After the phase shifts all pulses can be summed coherently to get the maximum response from each *point target.*

Figure 2.2 illustrates the idea with an orbiting SAR satellite. In the illustration the antenna's beamwidth in the azimuth direction is Θ_a that corresponds to the distance W on the ground. If the satellite's orbital velocity is v_s, it will take $t_r = W/v_s$ time units to travel the distance W on the orbit ($W \ll R_E$ and the orbit can be approximated with a straight line). As the footprint moves the same distance on the ground, the orbiting SAR can get measurements from each point target during the time t_r.

The range resolution of the SAR is the same as that of the conventional real aperture radar. The azimuth resolution, however, differs due to the SAR processing. To derive the resolution we examine the imaging configuration in Figure 2.2 by using the *synthetic array approach.* Firstly, suppose that we have an antenna of the length L that is long enough in order to the antenna's beamwidth to be small. Now, the phase and the intensity of the backscattered electromagnetic wave are measured at every $\Delta t = L/v_s$. The phase difference between measurements from two adjacent positions on the satellite's track is given by

$$\Delta\phi = 2kL\beta, \tag{2.8}$$

where k is the *wave number* and β is half of the beamwidth. When the phase difference reaches π, the two received waves are in total interference and will cancel each other. The cancellation point also corresponds to the first null in the antenna's directional pattern. Thus, the half beamwidth becomes

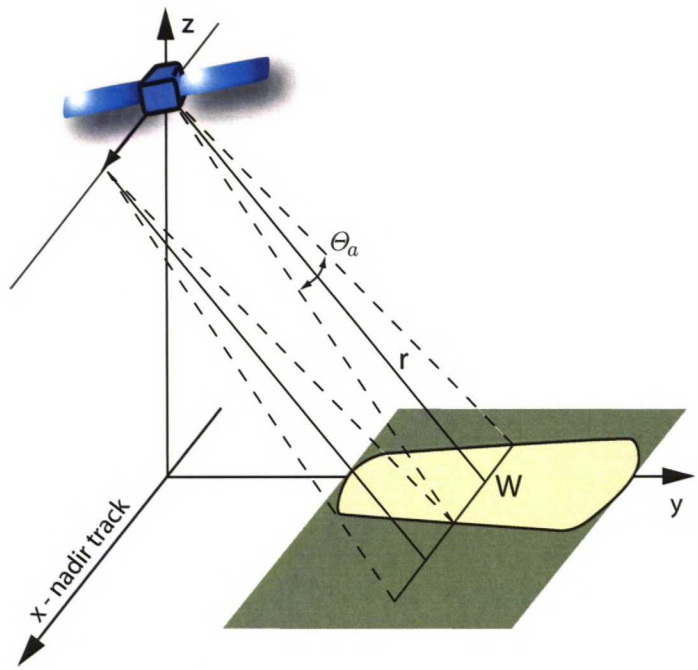

Figure 2.2: SAR imaging configuration. Θ_a is the antenna's beam width corresponding to the distance W on the ground and r is the range from the antenna to the swath's mid-point.

$$\beta = \frac{\pi}{2kL}. \tag{2.9}$$

Inserting the wave number $k = 2\pi/\lambda$ to Equation 2.9, we can express the total beamwidth for the synthetic aperture antenna by

$$\Theta_s = \frac{\lambda}{2L}, \tag{2.10}$$

which is exactly half of the beamwidth Θ_a of the real aperture antenna having the same physical length. For the azimuth resolution we get

$$\rho_x = \frac{\lambda r}{2L}. \tag{2.11}$$

Because a single target stays inside the footprint during the time t_r, the maximum length for the synthetic aperture is the distance W that is travelled on the track during that time. By using relations $W = \Theta_a r$ and $\Theta_a = \lambda/D_a$, we can re-write Equation 2.11 to

$$\rho_x = \frac{D_a}{2}, \tag{2.12}$$

which is the best possible azimuth resolution achievable by the synthetic aperture radar. In

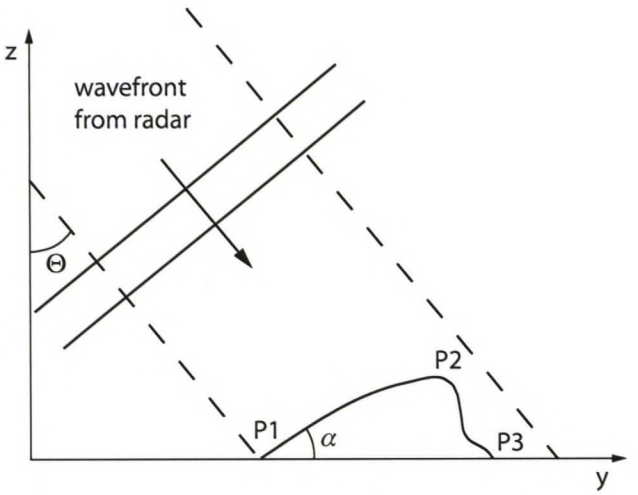

Figure 2.3: Foreshortening and shadow effects in a slope. Θ is the incidence angle, $P1-P3$ are the points in different parts of the slope and α is the constant for steepness.

essence, the finest azimuth resolution we can obtain is exactly half of the physical length of the antenna that is used in a radar. Comparing the azimuth resolution of the real aperture radar (see Equation 2.7) and of the SAR (see Equation 2.12), it can be noted that the latter is independent of a target's distance. Theoretically, for the SAR the halving of D_a doubles the resolution, conversely than for the real aperture radar. Finally, the end product after the SAR processing is an image strip that needs to be re-sampled to a square grid, likewise in the case of the real aperture radar.

The image strip is subsequently interpolated to the square grid whose pixel size corresponds to the maximum resolution cell in the data recorded by the SAR. Further, the image still has geometric distortions due to the motion of a radar platform and the side-looking configuration that separates the image's pixels by their time delays. Figure 2.3 depicts a typical situation in which a propagating wavefront bounces back from the different parts of a hill.The time delays from points $P1$ and $P2$ in Figure 2.3 have only a slight difference though their distance on the ground is notable. With the points $P2$ and $P3$ the situation is vice-versa. This phenomena is *foreshortening* that distorts the radar image so that a slope toward the wavefront is shortened and the other side is stretched.

Another phenomena manifesting itself in Figure 2.3 is *shadowing*. The area between points $P2$ and $P3$ is disguised from the wavefront due to the steepness and thus it will not appear in the radar image. The third effect by the terrain is *layover* that occurs when the angle of the slope toward the wavefront outruns the incidence angle, i.e. $\alpha > \Theta$. In that case the occurrence of $P1$ and $P2$ in the radar image gets mixed and the upper part of the

hill folds over the lower part. Both shadowing and layover cause a loss of information in the result image and cannot be corrected. Foreshortening can be corrected with the *digital elevation model* (DEM) of an imaged area. The correction is done along with the *geocoding* process that lines up the image to a desired map projection like UTM-35, for example.

An important factor that degrades the image quality in all *coherent imaging systems*, e.g. in laser, radar, sonar or ultrasound systems, is *speckle*. The term *coherent imaging system* means that the response from the ground is a vector sum of responses from all discrete scatterers inside one resolution cell. Denoting the amplitude of the scattered wave by A and the phase change by ϕ, we get the standard complex representation $Ae^{i\phi}$ for an electromagnetic wave. If inside a resolution cell there are N discrete scatterers we can express the total return by

$$Ae^{i\phi} = \sum_{k=1}^{N} A_k e^{i\phi_k}, \tag{2.13}$$

where A_k and ϕ_k are the amplitude and the phase of a single scatterer.

In practise, because of the distance between a SAR satellite and its target area compared to the wavelength (at C-band ca. 5.6 cm, for example), ϕ_k can be considered as equally distributed $[-\pi, \pi]$ and independent from A_k. Hence, the total return expresses the vector sum of a random walk in the complex plane. To put it simply, speckle implies that homogeneous regions on the ground, having similar scattering properties, are in the radar image disturbed by the fluctuation of pixels' values. For scatterers with similar properties it can be shown that the resulting amplitude A is *Rayleigh-distributed*

$$P_A(A) = \frac{2A}{\sigma} \exp\left(-\frac{A^2}{\sigma}\right), \tag{2.14}$$

where $A \geq 0$, σ is average intensity and mean and standard deviation are $\frac{\sqrt{\pi\sigma}}{2}$ and $\sqrt{\left(1 - \frac{\pi}{4}\right)\sigma}$, respectively. However, satellite SARs typically observe the received wave's intensity, which is $I = A^2$ and has a *negative exponential distribution*

$$P_I(I) = \frac{1}{\sigma} \exp\left(-\frac{I}{\sigma}\right) \tag{2.15}$$

instead of Rayleigh distribution. In Equation 2.15, $I \geq 0$ and mean and standard deviation are both equal to σ that is the average observed intensity. Both Rayleigh and negative exponential distribution imply that the variation in pixels' values is high because of the large tails of these distributions. The variation, in turn, is perceived as noise on a region having pixels with an approximately equal mean value.

A standard method to reduce the effect of speckle is *averaging* or *multilooking* the

intensity SAR image. In the process a SAR image is re-sampled so that the average of L (typically 3 or 4 with spaceborne SARs) adjoined pixels forms a new pixel. In mathematical terms the process corresponds to forming a *maximum likelyhood estimate* (MLE) for a pixel by using L independent observations I_k of the intensity according to

$$I = \frac{1}{L}\sum_{k=1}^{L} I_k. \tag{2.16}$$

However, in order the new pixel's value to be a valid MLE, it must be assumed that the RCS stays constant over the L pixel area that is averaged. Another fundamental issue is that the L-look average intensity image no longer obeys the negative exponential distribution shown in Equation 2.15. Instead, it can be analytically proved that multilooking results in a Gamma distribution

$$P_I(I) = \frac{1}{\Gamma(L)}\left(\frac{L}{\sigma}\right)^L I^{L-1}e^{-(LI)/\sigma}, \tag{2.17}$$

where $I \geq 0$. Further, in Equation 2.17, σ and L are mean and order parameter (or the number of looks), respectively. In statistical terms, the advantage of a multilooked image over a single-look one is that the variance is reduced by a factor L, i.e. by the number of independent measurements. Specifically, the variance in Equation 2.15 is σ^2, whereas in Equation 2.17 it is σ^2/L. Obviously, at the same time the resolution of the SAR image is degraded by L. It is worth of noting that the presented Gamma distribution is in a key role both in the Active Contour Model and in the de-speckling method which are explained in Sections 2.3.2 and 3.

For the distribution purposes SAR images are usually calibrated so that each pixel's value corresponds to the multilooked intensity that is taken a logarithm (also known as σ^0-*image*). The aforementioned pre-processing step can be though as *standardisation of variance*, because the variance in a σ^0-image no more depends on the mean, unlike in a normal intensity or in an amplitude image. Having a calibrated image, all to do to complete the preprocessing is to correct geometrical distortions and geocode the image to a desired coordinate system and map projection. The end product after the pre-processing is called *scene*. The SAR scenes used in this study are in the UTM-35 map projection (of the Northern Hemisphere) and use the WGS-84 coordinate system.

2.1.2. Open water in SAR images

In the following the appearance of water in SAR images will be considered especially from the flood detection's point of view. However, before that we take a closer look how water manifests itself in the SAR image in normal conditions.

The term *open water area* is here defined to include all natural and human-made water areas like seas, lakes, artificial lakes, ponds, water channels, rivers, etc. When an open water area is still, i.e. there's no wind, it can be modelled as a smooth surface that has *specular scattering* properties. In practise the specular scattering means that most of the energy sent by the radar is reflected away from the direction of the radar's receive antenna. Thus, open water areas yield low backscattering values and appear in SAR images as areas dominated by dark tones. [8, 9]

Because of the wavelength of the satellite SARs, the surface does not have to be perfectly smooth for low backscattering returns. The *Rayleigh criterion* for a smooth surface is given by

$$h < \frac{\lambda}{8 \cos \eta}, \tag{2.18}$$

where h is the RMS roughness of the surface (compared to a perfect mirror surface) and η is the incidence angle [10]. For example, the ERS-2 SAR satellite [11] of which $\lambda = 5.6\,cm$ and $\eta = 23$ degrees will see a surface of which $h < 0,8\,cm$ as flat. However, as the surface gets more rough the backscatter returns start to increase. In practise the water areas roughened by wind, current or heavy rain can resemble closely to dry land areas [12, 13, 14]. The phenomena is especially notable in VV-polarised images whereas HH-polarised images generally provide better contrast between land and water. The contrast also depends from the incidence angle. The larger the angle, the poorer the contrast between land and water and vice versa [14].

During flood conditions, however, the specular reflection model for water areas is no more usable. Flood covered areas, or *inundated areas*, may have emergent vegetation like crops, trees, plants, shrubs and twigs that cause increased backscattering. There are two mechanisms that yield increased returns compared to the specular scattering: the *volume backscattering* coming from canopy structures and the *double-reflection* or the *double-bounce* from trunks and twigs (see Figure 2.4). The ratio of the backscattering returns of these two mechanisms depends from the wavelength, polarisation and incidence angle. At L- and P-band the canopy penetration properties are better yielding double-reflection dominated returns, whereas at C-band the situation is vice versa. For VV polarisation the contribution of canopy volume backscattering has been found to be greater than for HH polarisation. Similarly, for low incidence angles (> 35 degrees) the contribution from canopy is higher than for high angles (< 35 degrees). For a large variety of different forest types the increases in backscattering are reported to be between 3 and 10 dB. [8, 9, 15]

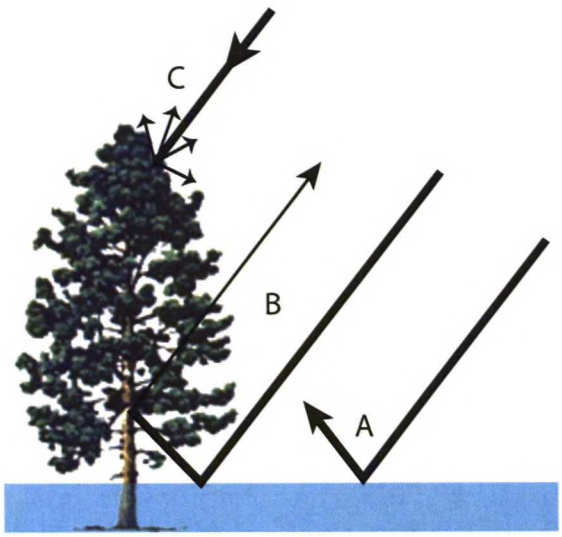

Figure 2.4: Backscattering mechanisms in an inundated vegetation. A represents specular scattering, B double-reflection and C the volume scattering from canopy.

2.2. Flood detection methods

As described in Section 2.1.2 the inundated areas in SAR scenes may manifest themselves either by increased backscatter or by near-to-zero backscatter. For example, an inundated forest's pixel value distribution is far away from the distribution of the flood plain without emergent vegetation. In a restricted case one can use exactly this difference to classify pixels to *non-flooded* and *flooded* classes. But when the *all* flooded areas with or without emergent vegetation are to be classified, the task turns out to be much harder. In the following we will briefly review some historical flood detection studies.

Traditionally open water areas have been detected in SAR images by using the *density slicing* method (other names include *slicing* and *thresholding*). The density slicing stands for classifying an image to desirable classes pixel by pixel, depending on the pixel's value. The method's name comes from the procedure in which class boundaries (i.e. min and max value for the pixels belonging to a class) are assigned by slicing the image's histogram vertically. As a result, the histogram is divided to M non-overlapping blocks which correspond to M different classes. The class boundaries can be set empirically or by using an unsupervised or a supervised algorithm dedicated to the task. Normally both kind of algorithms require at least the number of classes M as an input. Considering the flood detection problem, the number of classes is well known. Most often the desired classes are *non-flooded* and *flooded* or *water* and *land*. Further, unsupervised algorithms search for M *linearly separable* groups in the pixel value space. If the pixel value space contains more

than one dimensions like in a polarimetric radar image, for example, the algorithm's result is boundaries for M subspaces. Unlike the unsupervised algorithms, the supervised ones require training samples from each class. The training samples are small areas belonging to a certain class. After a supervised algorithm has been told which sample belongs to which class, it is able to compute the class boundaries.

Imhoff et al. [16] used the density slicing method to classify the SIR-B[2] radar's L-band image of a monsoon flood in Bangladesh to village, agriculture, water and river vegetation classes. The class boundaries were assigned empirically by hand, based on the ground truth information. The accuracy was assessed by first classifying the test locations, which formed a regular grid in the SIR-B image, and then determining the correct classifications by using an aerial photograph as a reference. The SIR-B image was acquired on 11th October 1984 and the aerial photograph on March 1983. Because of the one year offset, the reported water classification accuracy of 85 % can be regarded only as a rough estimate.

A frequently used supervised algorithm for setting class boundaries is the *decision tree* method. In essence a decision tree consist of a start point called *root node* and its child nodes which have they own child nodes, etc. Finally, each branch is terminated by a *leaf node* that corresponds to a class. Other than the leaf nodes contain a *rule* according which the decision to go to a child node is made. A special case of the decision tree is the *binary decision tree* (BDT) whose nodes have exactly two child nodes. An example of the binary decision tree is given in Figure 2.5. In the example, only single channel data is used to classify a pixel to the water class or to the land class. Pixels whose backscattering coefficient is less than -18 dB are classified to the water class and other pixels to the land class.

The construction of a decision tree can be done in various ways, depending on what parameter (e.g. misclassification rate) we want to optimise. In case of flood mapping, a statistical software package called S-Plus has been used at least by Townsend [15, 17] and Hess [18, 19] to build decision trees. Townsend used the decision tree to classify forested areas nearby a river to non-flooded and flooded classes. In eleven C-band, HH-polarised Radarsat scenes, which were acquired acquired in 1996-1998, training sites from flooded forest and upland forest that never floods were selected. The training data were input to S-Plus for building a decision tree. Further, in [15] Townsend used 11 wells (water gauges) to determine non-flood and flood conditions. In [17] total 202 test sites were selected and ground measurements were obtained from those sites. The ground measurements included e.g. a visual inspection and a water level measurement. In the both studies the reported classification accuracy of the test sites ranged from 90 to 98.1 % depending on the Radarsat

[2]SIR-B was NASA's Shuttle Imaging Radar mission that flew in 1984. For more information, visit at `http://southport.jpl.nasa.gov/scienceapps/sirb.html`.

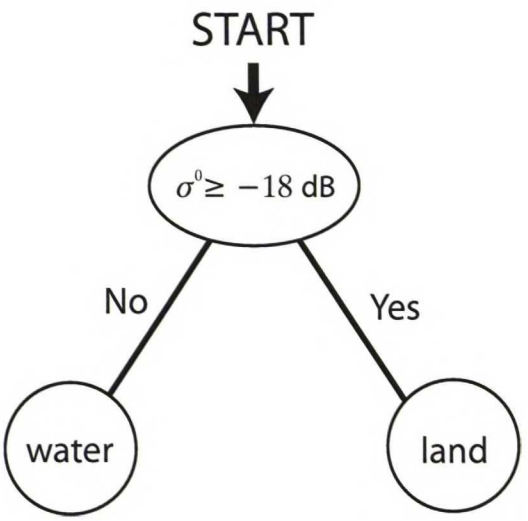

Figure 2.5: An example of the binary decision tree. The tree uses only one variable (σ^0) to classify a pixel to the land or the water class.

scene.

Besides Townsend the decision tree method has been used by Hess [18, 19]. In [19] Hess studied the mapping of the Amazon river floodplain by using two SIR-C[3] multifrequency, polarimetric SAR scenes. Total 8 radar parameters including σ^0 of HH, VV and HV polarisation and HH-VV phase difference all both at C- and at L-band were used. The ground truth data consisted of water level measurements, visual observations and photos taken from a low-flying aircraft. Total 55 and 60 test sites were selected for the first and the second SIR-C scene, respectively. For these test sites the reported classification accuracies ranged from 95.6 to 100 %, for water, non-flooded forest and flooded forest classes.

Considering the studies carried out by Imhoff, Townsend and Hess, it is worth of noting that all they used discrete test sites for the validation of classification accuracy. For example, Townsend's 11 water gauges in an 114 x 92 km study area [15] can at most provide very limited information about the flood plain's real coverage. Hess's 60 test sites for an 18 x 73 km study area [19] do not give any more convincing picture about the results, though being a somewhat denser grid than the Townsend's. For real classification accuracy validation it would be important to have a reference that covers the whole imaged area in order to be able to use *all* pixels for the validation. Further, the reference should temporally represent the situation at the moment of the image acquisition. The last point made about Hess's study [19] is that usually the more imaging channels (i.e. polarisations)

[3]SIR-C was NASA's Shuttle Imaging Radar mission that flew in 1994. For more information, visit at `http://lpdaac.usgs.gov/sir-c`.

whose correlation is small can be used, the better chances we have to *linearly* separate the classes whose distributions would overlap in fewer dimensions.

Having briefly reviewed the examples of the flood detection by density slicing, it is necessary to remind about the special properties of the SAR image before proceeding to other flood detection methods. Unlike in the optical image, the SAR image exhibits the speckle property, as explained in Section 2.1.1. For density slicing speckle may cause problems in the form of misclassified pixels. The misclassification results directly from the fact that due to speckle the distribution of a water area overlaps with the distribution of a non-water area. Consequently, a *water mask* may be fragmented and contains holes that are caused by speckle. To counteract this effect, a SAR image is usually *de-speckled* with a filter that is designed to reduce the variation in pixels' values over a homogeneous area. The drawback is that the more we smooth the SAR image, the more we lose contrast between land and water areas. Thus, one must always make a trade-off between smoothing and preserving contrasts. A variety of different speckle filters has been developed, from simple averaging filters to far more advanced ones that have a built-in model for speckle, for example.

Speckle is not the only problem in the detection of water areas. As noted in Section 2.1.2, open water areas can be roughened by wind or by rain that both cause more severe effects than speckle. The rough surfaces result increased backscattering and thus bright spots on open water areas that otherwise appear as dark regions in the SAR image. Usually this kind of spots appear on open sea and on open of big lakes because of the wind conditions there. However, it is worth of noting that wind can cause roughening also on a flood plain on a crop field, for example.

Another source of confusion for flood detection methods are smooth, man-made objects. These can be, for example, runways at airports, highways or large, flat roofs that all result near-to-zero backscatter in SAR images. Indeed, these smooth objects cause exactly an opposite kind of classification error than wind or rain. Consequently, without prior knowledge about either the man-made smooth objects or the water bodies, it is impossible to avoid aforementioned classification errors.

Because of the pixel-wise density slicing's sensitivity to problems caused by speckle and rough surfaces, other methods like *window-based* or *windowed classifier* and *active contours* have been developed. In the following we will briefly consider some examples of these algorithms. A window-based classifier uses a 8 x 8 pixels area, for example, that centred around a pixel to be classified. Further, the window's pixels are used to compute local statistics (or *features*) for the classification. Hence, the window's texture will affect to the classification that is no more based on a single pixel's value. An example of a window-based classifier is the texture surface water detector developed by Solbø [14]. He

uses a 5 x 5 pixels window for computing a *feature vector*

$$\overline{s} = \begin{pmatrix} |\max(x) - \min(x)| \\ \mu_x \\ v_x \end{pmatrix}, \tag{2.19}$$

where x belongs to the set of pixels inside the window. These features are subsequently input to a *neural network* that is trained by the backpropagation algorithm with randomly sampled training pixels over a SAR scene. Further, the result of a pre-classification is used for assigning correct class labels to the training samples. The pre-classification, in turn, is done with traditional thresholding by using a threshold value that minimises the least mean squares (LMS) *cost function*

$$\varepsilon(t) = \sum_{i,j} (f_t(i,j) - l(i,j))^2, \tag{2.20}$$

where $f_t(i,j)$ is the class label assigned to the pixel (i,j) after the thresholding by t and $l(i,j)$ is the correct class label obtained from a digital map. In essence, the trained neural network corresponds to a maximum likelihood (ML) classifier with Gaussian class probabilities. Solbø used two Radarsat SAR scenes (modes S2 and S7)[4] from same area for testing the surface water detector. Reference water masks were derived from digital maps. Reported classification accuracies for a full scene ranged from 97.2 to 98.4 % for the land class and from 76.1 to 79.9 % for the water class. In contrast to the previously reviewed studies, Solbø [14] used a full SAR scene for the validation of the classification accuracy. However, the accuracy of the digital water contours from the digital maps was not assessed at all. Lastly, the detector's pre-classification phase requires a digital map over the area that is to classified.

Third type of flood detection method that has been previously used is the active contour method. Active contours or snakes are numerical algorithms that were originally designed for the image segmentation task in the computer science. Because snakes will be explained in detail in Section 2.3, their theory is omitted here. Instead, in the following we review some studies that have utilised a snake designed to find desired areas in the SAR image. For example, Horritt has used his own active contour[5] algorithm [20] for delineating flood plains in ERS-1 SAR images [12] and flooded vegetation in E-SAR[6] images [9]. Further,

[4]The modes S2 and S7 correspond to 23° and 45° incidence angles, respectively.

[5]From now on Horritt's method is written as "the Active Contour" to make a distinction to the general term "active contour".

[6]E-SAR is an airborne SAR capable of producing polarimetric data from different bands. In this case fully polarimetric L-band and dual polarisation C-band data were acquired. The acquisition altitude has

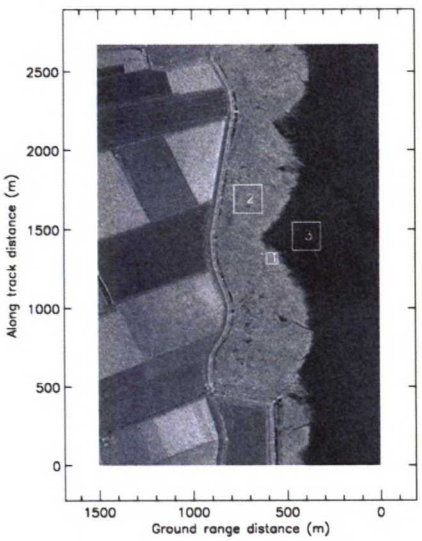

Figure 2.6: Study area with test spots corresponding to inundated vegetation (1), land (2) and open water (3). (Adapted from [9].)

Mason used his own snake algorithm for the delineation of shorelines in ERS-1 SAR images [21].

In [9] Horritt used a test area of 1.5 x 2.5 km (see Figure 2.6) in a 4-look, C-VV airborne SAR image having a pixel size of 1.5 x 2.5 meters. The reference water contours for the moment of the SAR acquisition were created by using DEM and LiDAR measurements. Specifically, the test image represents a shoreline dividing the image vertically to land and water regions. Because of the shallow beach, the transition from water to land is not sharp but in the between there is a zone of inundated vegetation. Hence, Horritt's idea was to segment the test image to three portions: land, inundated vegetation and open water. The distributions of these tree classes are given in Figure 2.7. For the segmentation two separate Active Contours were used, the other being initialised over a small land area and the other over a small open water area. After the convergence of each Active Contour, the resulting contours effectively separated the three aforementioned classes.

Reported classification accuracies for wet (inundated vegetation and open water) and dry (land) areas were 34.1 and 61.3 %, respectively. However, as the Active Contours were initialised by hand on areas that were known for sure to represent a certain class, the term *classification* is a bit misleading in this context. The Active Contour performs rather the *delineation* of a desired area than the classification of an image. Thus, we must have a

been about 3000 meters.

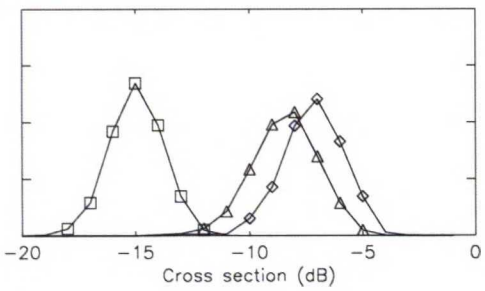

Figure 2.7: Empirical distributions of land (△), inundated vegetation (◇) and open water (□) areas corresponding the test spots in Figure 2.6. (Adapted from [9].)

priori information to select a proper initialisation area. In [9] the initialisation areas were selected by visual inspection.

Unlike in existing airborne SARs, satellite SARs have much poorer resolution that in turn reflects to the Active Contour's performance because of its statistical nature (see 2.3.2). Both Horritt [12] and Mason [21] have studied the use of active contours for mapping water areas in ERS-1 images. From these two Horritt's study that provides a quantitative accuracy evaluation will be reviewed briefly in the following.

Horritt used study areas of size 15 x 7 km and 14 x 12 km in River Thames in the United Kingdom. In total three ERS-1 C-VV scenes, of which one captured a flood event, were acquired from the study areas. The flood event was also captured in an aerial photo that timed within 6 hours of the ERS-1 scene that captured the flood. For delineating the flood plain an Active Contour [20] was initialised along a strip containing only the river's pixels. After the convergence the water contour was directly compared to the reference water line digitised from the aerial photo. More exactly, the distance between the water contour and the closest point on the aerial shoreline was computed. Based on the distribution of these distances, Horritt reported that about 70 % of the result shoreline was within 20 m from the reference line. Traditional classification results were computed by assigning pixels to the water and the land class by the water contour. Correct classification rates in the study area 1 were 22.6 and 67.9 % for water and land, respectively. Counting in the results from both study areas, in overall about 75 % of the area was classified correctly.

The reviewed seven studies that used three different type of classifiers provided a good cross-section of the flood detection in typical, challenging circumstances, mainly due to surface roughness Based on the observations made, we can conclude that the designing of a flood detector for all type of floods is not a simple task. Especially, the overlapping of the distributions of inundated vegetation and land (see Figure 2.7) create problems for flood classifiers. Based on the typical flood events in Finland and on the available spaceborne

SAR data, the flood classification problem was in this study bounded to detecting only open water areas by using C-band spaceborne SAR images.

2.3. Snake Models

In brief, *snakes* or *active contours* are algorithms that are capable of extracting desired borders from intensity or RGB images. With the *desired borders* we mean the accompanying of a priori information about the target areas to a snake algorithm in a form of initial values and constants. In other words, snakes are used to solve classic *image segmentation problem* with a priori information about the target areas. In addition to image segmentation, snakes have been used for tracking and matching of objects in image series or video clips.

Typically a snake consists of a directed, closed *polyline* that is represented by a series of nodes and an algorithm that drives the polyline to the borders of a target area. A visual example of a snake is presented in Figure 2.8 on page 27, in which the snake's nodes and segments, which link the nodes, are represented by red circles and red line segments, respectively. The direction of the snake is indicated by the green line segment, starting from the first node. Thus, in this case the snake is clockwise (CW) directed.

All snake algorithms are based on the *energy minimising scheme*, mainly differing on what kind of *energy function*, *minimisation algorithm* and *image* are used. Older snakes operated on the *edge* or the *gradient* image of the original intensity image, whereas newer ones use directly the intensity image. The energy function is usually formulated to include the terms corresponding to *image force* and *external constraints*. The former is computed from the image properties nearby the contour and the latter from the shape of the contour. By using the constraints it is possible to control the shape of the evolving contour and also include any a priori information to the energy function.

2.3.1. History

The first snake model was proposed by Kass [22] in 1987 and it utilised the techniques of variational calculus in the energy minimisation. The snake operated on the edge image that was computed from the original intensity image. The energy function proposed by Kass is

$$E^*_{snake} = \oint E_{int}(\mathbf{v}(s)) + E_{image}(\mathbf{v}(s)) + E_{con}(\mathbf{v}(s))ds, \tag{2.21}$$

where E_{int}, E_{image} and E_{con} are functions of the parametrised contour line $\mathbf{v}(s) \in \mathbb{R}^2$ that is, in turn, a function of the arc length s. E_{int} represents the *internal energy* of the contour,

i.e. its value is defined by the contour's curvature and bending. E_{image} is the *image energy* that is computed from the gradient strength over the contour in the edge image. Finally, E_{con} represents the *external constraints* that control the expansion of the snake and include a priori information. For the energy minimisation we have an $\mathbb{R}^2$ subspace, e.g. $x \in [0, N]$ and $y \in [0, M]$, whose area is equal to the one of the original intensity image and of the edge image. Having defined the search space, the core idea of Kass's method is to find the contour $\mathbf{v_{min}}$ that results the minimum value for $E^*_{snake}(\mathbf{v})$. As mentioned, Kass used variational calculus for this optimisation problem.

A year after the introduction of Kass's snake it was argued to be numerically unstable and have a tendency of nodes to bunch up on the contour, to the regions of strong edges [23]. Amini [23] proposed a *dynamic programming* energy minimisation scheme and a discrete grid (i.e. $\mathbf{v}(s) \in \mathbb{N}^2$) for the node positions. However, being a search-based method, the dynamic programming was very slow and memory consuming [24] with time and memory complexities of $O(nm^2)$[7] and $O(nm^3)$, respectively.

Compared to the *greedy algorithm*, that is used by modern snakes and that was introduced by Williams [24], both the variational and the dynamic programming methods lose in speed, robustness and simplicity. Namely, these *greedy snakes* running the greedy algorithm have time complexity of $O(nm)$ and memory requirement of $O(n)$. Unfortunately, the greedy algorithm cannot guarantee the convergence to the global minimum, like more sophisticated and computationally more expensive methods can do. Additionally, the greedy snake has some drawbacks of the early snakes pointed out in [24, 25, 26, 27, 28]. Two major problems are rooted to the formulation of the snake's internal resistance term. Firstly, the contour has a tendency to shrink if no attracting image force is present (that is, if the contour has not been initialised sufficiently close to the object boundary). Secondly, if the image force is present on some segments of the initialisation contour, the other parts of the snake are still unable to converge to the boundaries of the target object.

The convergence problem has been struggled by introducing various external force terms to the energy function in order to push a snake towards the borders when no image force is present. The *balloon snake* [25, 29] incorporates a constant force to the direction that is normal to the contour. However, the constant push or pull force is also included to the equilibrium state obliging the snake to settle a bit off from the real boundary of the target object. The *region force* is an improved version that is computed based on the values of a goodness function $G(x, y)$ and an area change δA [30, 20]. The goodness function drives a snake to the direction that both contains desired kind of pixels and where the area change is biggest. If all the pixels in the neighbourhood give same G values, the direction of the

[7] Here n is the number of nodes on the contour and m the number of possible locations that a node can move (thus, in a discrete grid m is usually 4 or 8). If not explicitly mentioned, all snake related complexity notations from now on will follow the aforementioned convention.

maximum area change is selected. Additional benefit is that the snake implementing the region force can operate directly to an intensity image, as the value of G is computed from pixels on the region enclosed by the contour. Consequently, these snakes are often referred as *statistical snakes*. An example of a statistical snake is in Figure 2.8 on page 27 that represents the snake developed by Horritt [20], converging to the boundary of an area containing Gamma-distributed noise. Horritt's snake will be covered in detail in Section 2.3.2.

To complete our review of the snake algorithms we briefly note *gradient vector flow* (GVF) snakes. In these snakes the image force has been replaced by the GVF field that is computed from the input image by minimising a desired energy functional [26, 27]. Advantages are the large capture range and the handling of concave or convex shapes. Compared to the greedy snakes, a clear disadvantage is considerably higher computational complexity.

2.3.2. Active Contour Model

The Active Contour model developed by Horritt [20] is based on the greedy algorithm [24] and the region force concept [30]. It incorporates a goodness function that is formulated with a priori experimental information about the distributions of the local mean and the local variance in SAR images. The Active Contour has been successfully used for flood boundary delineation from SAR images [12, 9]. In this section we will first present the mathematical definition of the Active Contour and then turn to explain how it works in practise by using an example that is presented in Figure 2.8 on page 27.

The Active Contour bases its energy minimisation scheme to the function given by Equation 2.22. The function is formulated by using a parametrised contour representation $\mathbf{u}(x(s), y(s))$, where s is the arc length and $\mathbf{u}$ itself is the contour's position vector. δE_{tens} and δE_{curv} are the respective changes in *tension* (see Equation 2.24) and in *curvature* (see Equation 2.25) when the snake is expanding or shrinking. These changes occur when a contour $\mathbf{u}_n$ is deformed to $\mathbf{u}_{n+1}$, where n is the number of iteration. Further, in Equation 2.22 G is the goodness function and $I(\mathbf{u}(s))$ is a set of image pixels that both are defined more exactly later in this section.

$$\delta E = \delta E_{tens} + \delta E_{curv} - \oint G(I(\mathbf{u}(s)))\delta\mathbf{u} \cdot (\frac{\partial \mathbf{u}}{\partial s})_{\perp} ds \tag{2.22}$$

As $\frac{1}{2}\oint \mathbf{u} \cdot (\frac{\partial \mathbf{u}}{\partial s})_{\perp} ds$ is the total area enclosed by the contour, the integrand in Equation 2.22 is the local area change (actually $2\delta A$) multiplied by the goodness function. This continuous case's formulation must be discretised in order to program the algorithm to a computer. The discretised version becomes

$$\delta E = \delta E_{tens} + \delta E_{curv} - G\delta A, \tag{2.23}$$

where each term matches exactly with the terms in Equation 2.22. However, the integral is approximated by $G\delta A$. Unlike in the continuous case, in Equation 2.23 nodes can have only discrete positions so that $\mathbf{u}(s) \in \mathbb{Z}^2$. Consequently, all factors in Equation 2.23 can have only discrete values.

The roles of δE_{tens} and δE_{curv} are to promote even node displacement and smooth contour by influencing to δE that is computed for each node's 8-neighbours during an iteration. In that way δE_{tens} and δE_{curv} affect to which 8-neighbour will hold the maximum negative change in energy and consequently will be selected to be the next position of the node. Further, δE_{tens} and δE_{curv} are computed by using the definitions of the *total tension energy* and the *total curvature energy* that are given by

$$E_{tens} = \lambda \sum_i \|\mathbf{u}_{i+1} - \mathbf{u}_i\|^2 \tag{2.24}$$

and

$$E_{curv} = \gamma \sum_i \frac{\|\mathbf{v}_{i+1} - \mathbf{v}_i\|^2}{a}. \tag{2.25}$$

In the latter equation

$$\mathbf{v}_{k+1} = \frac{\mathbf{u}_{k+1} - \mathbf{u}_k}{\|\mathbf{u}_{k+1} - \mathbf{u}_k\|}, \tag{2.26}$$

and a is the distance between midpoints of $\mathbf{u}_k - \mathbf{u}_{k-1}$ and $\mathbf{u}_{k+1} - \mathbf{u}_k$. In Equations 2.24 and 2.25, λ and γ are weighting constants for the respective measures. The discrete curvature measure in Equation 2.25 results the same range of values, which is $[0, \infty[$, than the integral definition of curvature in the continuous case. The discrete tension measure in Equation 2.24 results high values when two nodes wander further from each other. Hence, the idea of the tension measure is to keep adjacent nodes within the pre-defined nodal spacing by affecting to δE in Equation 2.23. However, the tension measure does not result higher values when two nodes come very close. Thus, an additional *re-parametrisation stage* is required to delete and add nodes to keep the nodal spacing within the pre-defined range. The importance of the nodal spacing will be discussed shortly when the goodness function is explained.

The purpose of the goodness function G is to indicate whether or not the pixels in a certain direction of movement of a node have desired statistics. More exactly, these statistics are sample mean and sample variance. As the Active Contour is initialised above

some area, the mean and the variance of that area are computed and saved to be used as *seed statistics*. After the initialisation the algorithm starts to loop all its nodes, computing the local mean and the local variance from a two-segment polyline $(\mathbf{u}_{k-1}, \mathbf{u}_k + \mathbf{d}_l, \mathbf{u}_{k+1})$, where k refers to a current node under processing and $\mathbf{d}_l$ is a unit vector for movement to k's each neighbour ($l = [1, 8] \in \mathbb{N}$). Pixels on the described two-segment polyline comprise exactly the set that is picked out by $I(\mathbf{u}(x, y))$ in Equation 2.22.

Horritt's idea was to formulate a goodness function that takes into account the distribution of pixels' values in SAR images and scores its argument pixels by using that information. The Central Limit Theorem is applied in formulating G_μ that is the sample mean part of the goodness function G. The Central Limit Theorem states that as the sample size $n \to \infty$, the mean computed from that sample is normally distributed. Hence, the sample mean part G_μ can be defined as

$$G_\mu(\mu') = 1 - \frac{n(\mu' - \mu)^2}{v\, k_\mu^2}, \tag{2.27}$$

where μ' is the sample mean, n is the sample size, μ and v are the seed statistics and k_μ is a user-selectable constant. Constraining the minimum value to -1, G_μ gives out values from the range $[-1, 1]$ that are distributed according to $\ln N(\mu, \frac{1}{2} v\, k_\mu^2)$. The core idea of the formulation is that the sample means matching closely with the seed mean will result goodness values that are near to 1. Further, the constant k can be used to manually control the shape of G_μ.

The distribution of sample variance is determined in [20] experimentally by taking samples from a Gamma-distributed population and then fitting a model to that data. In effect, based on those results the sample variances computed from a SAR image are fitted to a suitable Gamma distribution. Skipping some intermediate results, the sample variance part of the goodness function is

$$G_v(v') = \frac{1}{k_v^2} \left(\frac{-P\, v'}{\varphi} + P \ln \frac{P\, v'}{\varphi} - \ln v' \right) + C, \tag{2.28}$$

where v' is the sample variance, $P = P(n)$ is the *order parameter* and $\varphi = \varphi(v, n)$ is the mean sample variance, C is the constant that bounds G_v to 1 from above and k_v is a constant with the similar function as in Equation 2.27. $P(n)$ models the order parameter and $\varphi = \varphi(v, n)$ models the mean sample variance of the Gamma distribution of sample variances. Exact definitions of those functions can be found in [20].

Horritt's experimental tests showed that models for both the distribution of the sample mean and for the distribution of the sample variance in the SAR image hold sufficiently good when $n \geq 10$. Thus, for successfull delineation the nodal spacing s between two nodes must be ≥ 5 pixels implying the need of re-parametrisation as a snake evolves. The

re-parametrisation means simply deleting and adding nodes to keep the aforementioned condition valid. Having defined G_μ and G_v, the mathematical review of the Active Contour can be completed by defining the complete goodness function as

$$G = \alpha G_\mu + (1 - \alpha) G_v, \tag{2.29}$$

where α is the weighting constant for dividing the discriminatory effect between G_μ and G_v.

Before proceeding to the example it is worth of noting about two problematic issues with the Active Contour. First is how to find proper values for the algorithm's various constants. With the *proper values* we mean that by using those values the delineation result can be considered satisfactory. In [20] starting point values for λ and γ have been derived analytically. However, depending on the area to be delineated, these values may need to be adjusted along with k, α. Second issue is how to deal with intersecting snakes when delineating multiple areas in the same image. The setting of proper values for the constants and the handling of intersections are discussed in detail in Chapter 3.

Next we turn to examine an example of how the Active Contour algorithm works in practise. In Figure 2.8 there are six frames of 256 x 256 pixels each representing the evolving contour of an Active Contour. The purpose is to illustrate how the Active Contour can delineate a noisy area, even when the distributions of the target area and the background overlap significantly. The circle in each frame contains Gamma-distributed noise with mean $\sigma_{circle} = 120$ and variance $v_{circle} \approx 8313$. The background is also Gamma-distributed, but with mean $\sigma_{bg} = 100$ and variance $v_{bg} \approx 5774$.

The first step of the snake's operation is initialisation. In the frame in the upper-left corner the snake is initialised over a 100 x 100 pixels area in the centre. At this point the algorithm computes sample mean and sample variance from the area. To set the proper values for the snake's internal constants is not a trivial task, usually it requires some experimentation with given data. In this case the constants were: $\lambda = 0.4$, $\gamma = 20$, $k_\mu = 1.4$, $k_v = 1.3$ and $\alpha = 0.5$. The minimum and the maximum limit for the re-parametrisation were 25 and 30 pixels, respectively. In practise this means that a new node is added in the middle of a segment if its length exceeds 30 pixels. Oppositely, a node is deleted if its distance to either neighbour node drops below 25 pixels. To keep the snake intact, the deleted node's neighbour nodes are linked. In this example the re-parametrisation was executed after every 20th iteration.

After the initialisation the computation begins and the snake starts to expand over the regions containing pixels with similar statistics of those in the initialisation area. The expansion and the re-parametrisation can be noticed well in each frame starting from the first one. Theoretically a snake has found the global minimum of its energy function when

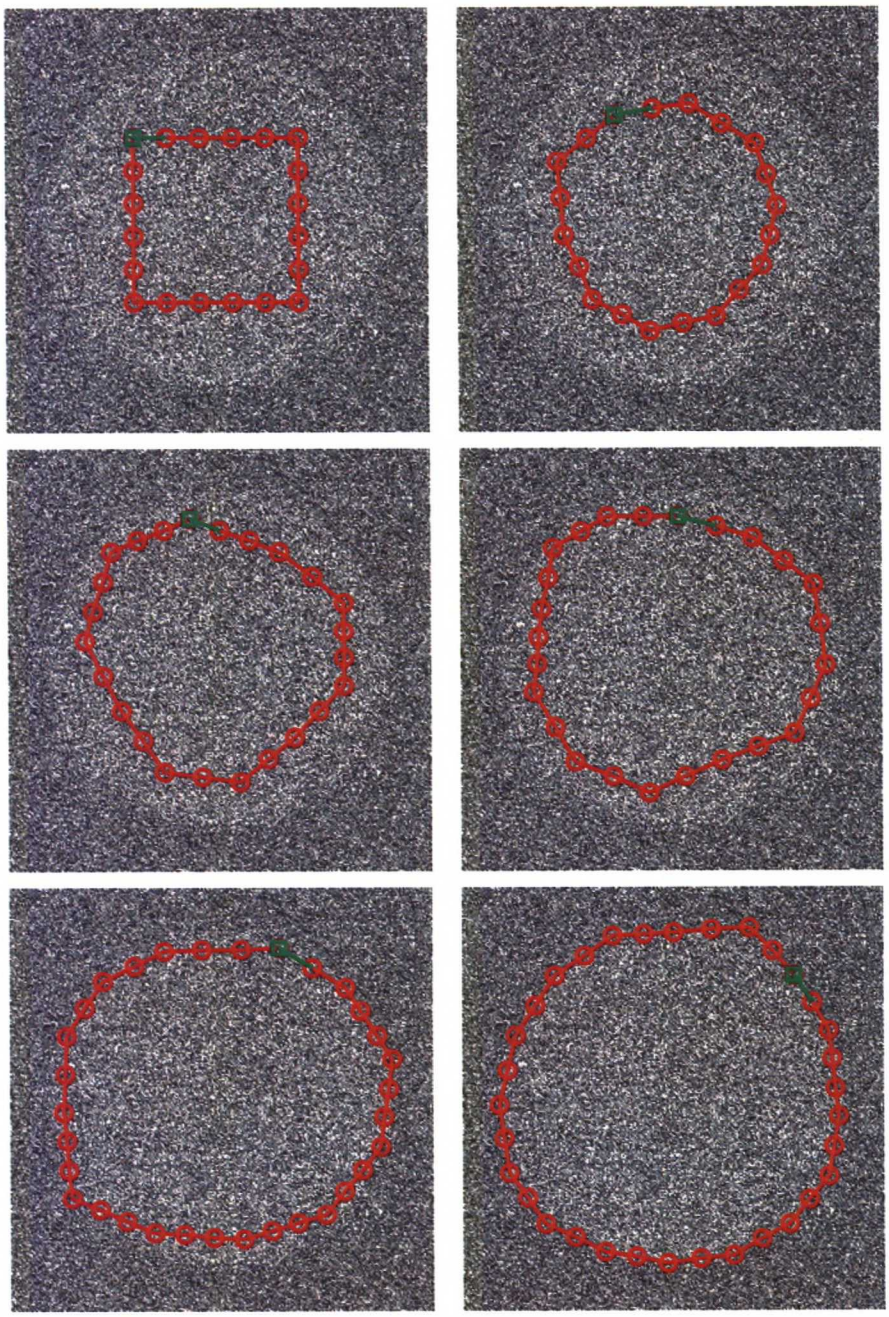

Figure 2.8: Snake converging to the boundary of a circle. The circle and the background are Gamma-distributed noise with means $\sigma_{circle} = 120$ and $\sigma_{bg} = 100$, respectively. The order parameter for both distributions is 3. The frames illustrate the snake's position after 0, 20, 40, 60, 80 and 120 iterations, starting row-wise from the upper left corner.

none of the nodes is moving. However, because nodes are permitted to have only discrete

positions, a node in a local minimum can get stuck to bounce between two neighbouring pixels. To counteract these situations an area convergence condition was created. Namely, as the snake's area does not expand significantly it has in most cases found the target area and can be frozen. The area convergence method computes a moving average (MAV) over the snake's extent during last 10 iterations. After each iteration i a MAV area A_i is compared with the previous value A_{i-1} and if $A_{i-1} \leq A_i$ does not hold, the snake is frozen. In the example the area convergence condition was fulfilled after 120 iterations. The result snake can be observed to approximate fairly well the original circle, in spite of the overlapping distributions of the circle and the background.

Chapter 3

Development of an automatic flood delineation system

The initial goal was to develop *a stand-alone program*[1] that could be integrated to Floodman project's *production line* and that could *automatically* delineate open water areas from SAR scenes. Other main specifications included that the program must be able to read some common raster file format that is used for storing SAR scenes and it must be able to produce some common vector file that contains the delineated water areas.

To understand better the design choices that were made along the development process, we first take a look to Floodman's production line, because in the end it was the SW platform above which the program had to be designed. The production line, which was developed by Floodman partner *NORUT IT*[2], is a piece of *IDL programming language* code for creating a graphical user interface and for starting external processes. The basic idea is that a user can select the flood detection method and the input SAR scene via the interface. After the user interaction the production line starts, or *spawns*, the selected method with the given input data. Subsequent actions will proceed automatically until the resulting water mask is stored to a file and the processing completes.

Having described the role of the production line, the focus is now returned to other issues in setting up the development environment. At the time of the beginning of Floodman project there were no implementations available of the Active Contour algorithm in any programming language. Thus, the only choice was to implement it from scratch, by using Horritt's article [20] that described the Active Contour's theory. Implementing the algorithm with a *low-level* programming language like *C* [31] would have automatically

[1] In essence, a stand-alone program is a binary file that can be executed without the need of any supporting applications.

[2] The company, *Norut Informasjonsteknologi*, is located in Tromsø, Norway and its web pages can be found at `http://www.itek.norut.no`.

resulted a stand-alone application, because all C programs need to be compiled from the source code to the binary file before the execution. However, the required time for the SW developing would have been tremendous, because no mathematical functions nor toolboxes appropriate for the C language implementation of the Active Contour were conveniently available.

The next alternative was the commercial *MATLAB*[3] scientific computing environment providing a *high-level* programming language. MATLAB has a large built-in set of functions for mathematical computation, image processing, data visualisation, etc. for easy and fast building of experimental algorithms. Normally all the built-in functions along with the user-written functions are *interpreted* by MATLAB. However, MATLAB also has a built-in compiler that produces stand-alone applications from MATLAB source code (or from *M-code*). The compiled programs can run in any computer without the need of the expensive MATLAB application licence. On the opposite side, the stand-alone applications compiled from M-code are considerably slower than same implementations in the C programming language. Nonetheless, because the execution speed was not the first priority, MATLAB was chosen to be the implementation language, in hope for reducing the development time.

The initial plan was first to develop the Active Contour SW for delineating a single area and then add more functionality. As stated in Section 2.3.2, the Active Contour requires that its polyline is initialised over the area containing desired kind of pixels. Thus, as our goal was to implement an *automatic* application for the flood delineation, we had also to develop an automatic method for the initialisation. The initialisation and the delineation can be considered as distinct steps, though the latter is dependent from the former. Consequently, to make the development and the testing easier, the program was divided to two steps: *pre-processing* and *delineation*. The pre-processing step includes de-speckling of the SAR scene, detection of the water areas and initialisation of the Active Contours. The delineation, in turn, runs all Active Contours until they have converged and finally outputs the resulting open water area mask. In the pre-processing step input parameters are the SAR scene and the configuration file, as illustrated in Figure 3.1. After the pre-processing step is completed, the initialised Active Contour data structures are saved together with the SAR scene to the *intermediate result file* that, in turn, is input to the second step along with its configuration file. Both configuration files contain all user-specified constants that are needed to complete the respective processing step.

Afterwards the whole SW development process can be seen to contain four distinct phases. The development progress from a phase to the next one is illustrated in Figure 3.2. The figure's *progress diagram* visualises the start point as a solid black circle with

[3]For more information, look at`http://www.mathworks.com/products/matlab`.

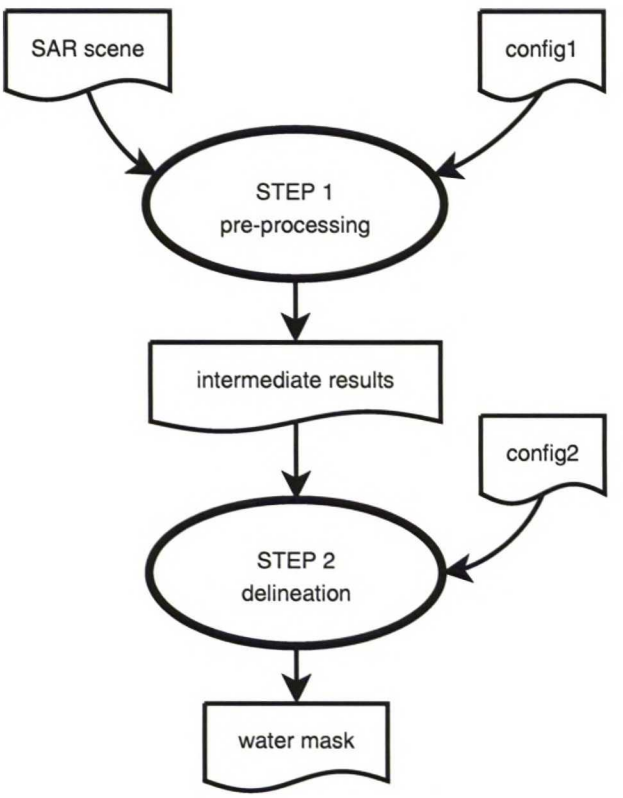

Figure 3.1: The two distinct programs that together complete step-wise, automatic open water area delineation. The rectangle-like boxes with curved lower border represent data files.

an arrowed line to the first phase that includes an implementation of the Active Contour algorithm for the delineation of a single area. In the rest three phases more functionality is added part by part, until the final implementation, which is marked by two coinciding circles, is ready. It is worth of noting that the development progress was not unambiguously linear but rather required iterative coding. Namely, in some cases adding a new function to the system required that already written functions had to be modified. Similarly, even a modest change to already defined data structures sometimes caused a bunch of changes to the existing functions. Further, the testing and the debugging of each new piece of code resulted usually minor refinements to the existing code. These iterative parts of the progress are depicted in Figure 3.2 by the slashed lines, recurring back to the previous phases. The rest of this chapter is divided into four sections which will cover the whole SW development process, phase by phase.

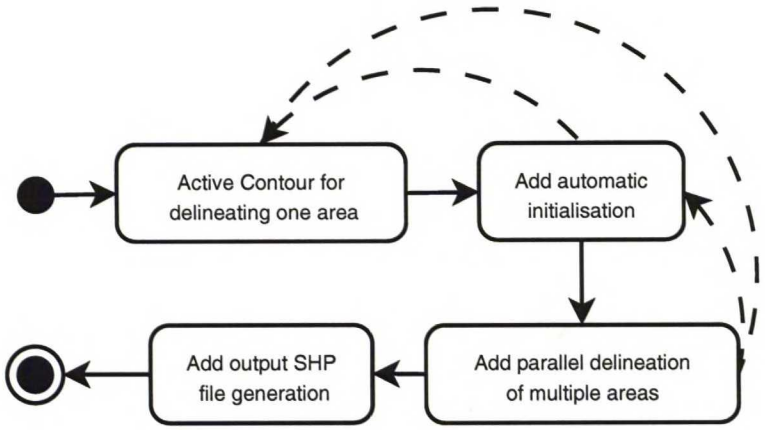

Figure 3.2: The development progress diagram. The start point is marked by a solid black circle and the end point by two coinciding circles. The slashed lines recurring back imply iteration during the development.

3.1. Single flood area delineation by using Active Contour

In this first development phase the Active Contour algorithm was implemented with a manual contour initialisation. The main function `mainboa` along with all other functions for this phase are visualised in Figure 3.3 on page 34 as a *dependency tree*. Briefly, the `initboa` function handles the initialisation of a snake and `runboa` computes new positions for all nodes during each iteration step. The runboa function is called by `runboa_h` which, in turn, takes care that the snake is re-parametrised by `reparamlist` as it expands. Further, `runboa_h` detects when the snake is converged according to the selected convergence criterion.

The most important function, however, is `deltaE` which computes ΔE (Equation 2.23 on page 24) for moves to each neighbour pixel of a node. Because the `deltaE` function is called nearly all the time the Active Contour algorithm is running, optimising its execution speed was an issue of the utmost importance. In the very first version of the `deltaE` function ΔEs for the neighbour pixels were computed in a `for`-loop, one by one. Afterwards in the first tests it was noted that the performance of the `for`-loop was indeed too slow for running multiple snakes in a SAR scene. Even with a single snake consisting of tenths of nodes the total execution time was too long. Luckily the solution was found easily from MATLAB's native support for the vectorised computation by basic math operators like `*`, `-`, `+`, and `^`. After converting the `for`-loop to a vectorised code, the performance of the program sped up by several factors.

Still the execution time of `deltaE` was not yet at the level that would provide enough speed for processing a full SAR scene. Thus, the time spent in `deltaE` and its subfunctions

constant	description
N_PRI1	Execution priority 1
N_PRI2	Execution priority 2

Table 3.1: Node's priority levels.

were logged by using MATLAB's built-in `profile` function. It turned out that most of the time was spent in the `improfile` function that was used to get the pixel values from the two-segment polyline connecting a node to its immediate neighbour nodes. When the `improfile` function's source code was inspected, it came obvious that the numerous input parameter checks for determining the desired usage of the function caused the slowness. Further, most of the features provided by `improfile` were not needed in the Active Contour implementation. Consequently, `improfile`'s source code was stripped off from all unnecessary parts in order to increase its execution speed. The new version, which is also shown in Figure 3.3, was then saved and renamed to `improfile_boa`.

One essential part of implementing of the first phase was to run tests with synthetical SAR images. The first test was similar to the Figure 2.8's example that was explained in Section 2.3.2. The purpose was to experiment the convergence condition that Horritt had used in his implementation. To understand better the convergence process it is necessary to first understand the *states* that a node can have during its travel from the initial point to the final position. Every time after having created a node to given coordinates, the `initboa` function sets the node's state to `N_PRI1`. As shown in Table 3.1, `N_PRI1` corresponds to the first execution priority, i.e. a node is computed a new position every time it is encountered by `runboa`. `N_PRI2`, in turn, refers to the second highest priority setting so that a node is computed a new position only every tenth[4] iteration. Now, as `runboa` loops nodes it also keeps track whether a node has stayed on same location during `MAX_SAMEPOS`[5] successive iterations. If it has, the node's priority is lowered to `N_PRI2` and thus it is assumed to be nearly converged. The rationale for using the word "nearly" is that if the node X_p's neighbour nodes X_{p-1} and X_{p+1} change their locations, it may happen that re-computing ΔEs for X_p results a new location. Thus, the `N_PRI2` nodes are computed every tenth round and in case they are moved, their priorities are raised to `N_PRI1` again. Aforementioned process is illustrated in the `runboa` function's state diagram in Appendix's Figure 5.3 on page 66.

The strict definition for a snake's convergence is that its every node has stabilised and

[4]This value (10) was adapted from Horritt's Active Contour article [20] and was empirically checked to work properly with SAR images whose size ranged from small 100 x 100 pix. extracts to full SAR scenes.

[5]The constant's numerical value (5) was selected empirically based on the tests with real SAR data.

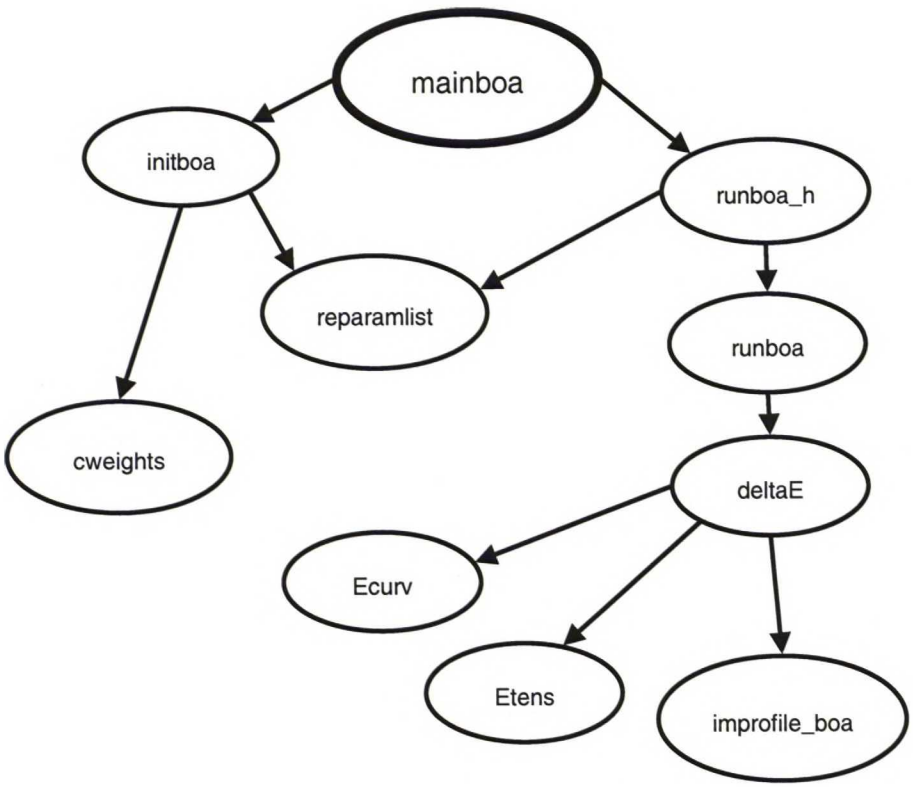

Figure 3.3: The dependency tree of the MATLAB functions implementing the SW development's first phase. A line with a single arrow indicates a one-way dependency.

thus the snake's line segments' coordinates do not change anymore. As referred briefly in Section 2.3.2, having discrete locations for the nodes and the greedy algorithm for the energy minimisation, the complete stabilisation is practically impossible to obtain. In the tests with clean water bodies in small SAR images it was noticed that usually the complete stabilisation was hindered by some nodes that were oscillating between two neighbouring pixels. To study further this issue, a piece of code logging the ratio of nodes in the `N_PRI2` state was written. Several small SAR images were delineated without any convergence condition, only limiting the maximum amount of iterations. A typical graph showing the ratio of `N_PRI2` nodes as a function of iteration number is given in Figure 3.4. In the beginning the ratio increases rapidly but then gets stuck to about 0.8 and remains there until the maximum number of iterations (in this case 200) has been reached. The stabilisation occurs around 45 iterations, implying that the snake's contour does not change anymore. As a result, the snake cannot satisfy the strict convergence condition, though its contour has practically locked to the border of the area. In other tests that were conducted the highest ratio of `N_PRI2` nodes ranged from 80 to 95 %. To counteract the

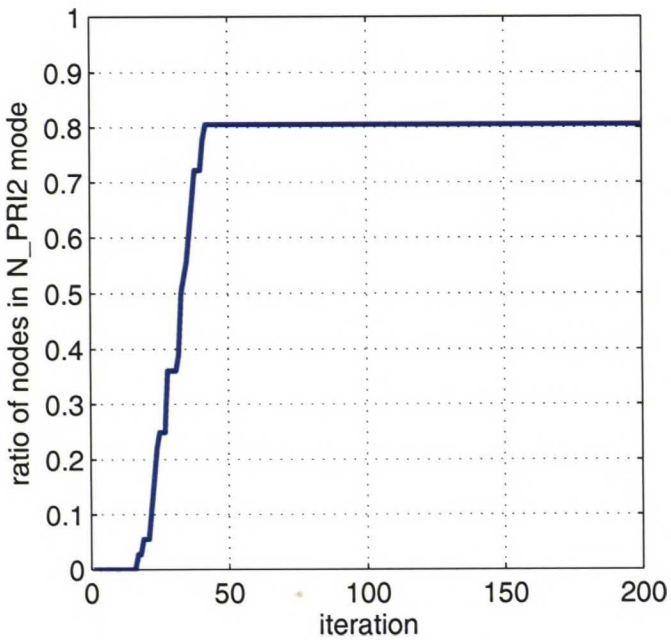

Figure 3.4: Typical graph of the ratio of nodes in the `N_PRI2` state as a function of the iteration number.

problem caused by the oscillations, the strict convergence condition was modified to allow the convergence after the ratio has reached the pre-defined value `PRI_CONV_LIM_RATIO`. Based on the tests, working values for the `PRI_CONV_LIM_RATIO` constant were found to variate between 0.85 and 0.9, depending on the SAR image.

3.2. Automatic initialisation

The second development phase for the system was to add the automatic initialisation of snakes. As mentioned in Section 2.3.2, the Active Contour needs to be initialised so that the pixels it encloses belong to the target area. Hence, the obvious problem was how to know a priori the location of a water body over which a snake could be initialised. One option we considered was the utilisation of the classification result from optical data covering the same area as the input SAR scene. For example, a pan-sharpened Landsat 7 [32] near infrared (NIR) image, which has a 15 x 15 meter pixel size, could be used as optical data. The reviewed studies [33, 34, 35] about the detection of water from optical data[6] concluded that especially the near infrared band was most suitable for the purpose. The thermal infrared (TIR) image has even more contrast between land and water but its

[6]The studies used optical data from Landsat, SPOT and EOS-Terra ASTER satellites.

resolution in currently operative optical satellites is noticeably poorer than the resolution of the NIR image. For example, the TIR resolution in Landsat 5 and 7 is 120 x 120 m and 60 x 60 m, respectively. Hence, the NIR band provides the best compromise between the resolution and the contrast between land and water. However, the water mask extracted from a NIR image is easily ruined by even minor clouds obscuring one or more water bodies. Further, the acquisition times of the optical and the radar image should match within few hours in order the optical data to be usable in the initialisation of Active Contour. If the optical and the radar image's acquisition times differ many hours, also the shorelines may differ considerably. That may the case especially during a rising or fading flood. Lastly, the need of an optical and a radar image to be able to classify the latter one is an unwanted restriction for the whole system that should aim for the near-real-time operation. Due to the aforementioned weaknesses, the option to use optical data for the initialisation was abandoned.

Another option was to use geoinformation system (GIS) data about existing water bodies. More exactly, bitmap landmasks or vector format shorelines were considered to be used in the initialisation of snakes over existing lakes, rivers, etc. After the initialisation the snakes would expand to flooded areas. However, the obvious limitation in the usage of GIS data is that only the inundated areas that are connected to the existing water bodies can be found. Thus, for example, an isolated flood plain, which could have been caused by e.g. heavy rain, on an agricultural field won't be detected. Additional restrictions include that the flood detection system could be used only in the SAR scenes for which suitable GIS data would be available. Since our goal was to design as autonomous a system as possible, the dependency of GIS data and its limitations in the initialisation were considered too restrictive and the alternative was cast aside.

The third approach was to pre-classify a SAR scene first with a fast method, resulting an approximate of the water mask that could be used for initialising the Active Contours. The traditional thresholding was tested with some SAR images and concluded to be a simple and fast method for finding most of the water bodies. Due to the speckle that is inherent is the SAR image, the thresholding applied directly to a SAR scene results extremely shattered water mask, containing a multitude of one-pixel areas. Thus, the SAR scene must be first de-speckled with a filter that performs enough smoothing still preserving well the contrast between land and water. Two different type of filters were compared; the other was a median filter and the other an adaptive statistical filter. The former represented a simple and computationally lightweight filter and the latter a sophisticated, complex and computationally far heavier solution. Consequently, the rationale for the selection was to see how these methods, which had quite opposite characteristics, would perform.

Before going to the comparison's results the selected filtering methods will be covered

briefly. The median filter simply sorts the values inside the window and then assigns the centre pixel the value that falls on the middle of the sorted values. For example, suppose we have a 11 x 11 pixels window containing in total 121 pixels. In that case the window's centre pixel is assigned the value of the 61st pixel in the list of sorted pixels. Consequently, the output of the median filter does not depend *linearly* on its inputs, unlike that of the mean filter, for example. The selected statistical filter is called *Gamma MAP* [3]. As it is named, it produces the *maximum a posteriori* (MAP) estimate for the underlying RCS by using prior knowledge about the Gamma distributions of the intensity and the RCS in the SAR image. In mathematical terms the *a posteriori* conditional probability for the RCS is defined by

$$P_{AP}(\sigma \mid I) \propto P(I \mid \sigma)P_{\sigma}(\sigma), \tag{3.1}$$

where σ is the RCS and I is the intensity. As noted in Section 2.1.1, for the L-look intensity image the conditional probability $P(I \mid \sigma)$ is given by Equation 2.17. Further, the second term in Equation 3.1 represents the a priori knowledge about the RCS's PDF that can be modelled as a Gamma distribution with mean μ and variance v. For estimating these statistics the Gamma MAP produces the mean and the variance images from the original image by passing a window over it. More exactly, the centre pixel of the window is assigned the local mean (or the local variance) computed from the pixels enclosed by the same window. The process is repeated until the window has been slided over the whole image, pixel by pixel. Having computed the estimates, the filter proceeds to find the MAP estimate σ_{MAP} for each pixel by maximising $ln(P_{AP}(\sigma \mid I))$ over σ. In practise the finding of the MAP estimate for a single pixel takes to solve a quadratic equation. Consequently, the Gamma MAP method's computational complexity is considerably higher than the mean filter's.

The comparison between the mean and the Gamma MAP filters was done by de-speckling a test ERS-2 SAR image. The image was already 3-looked and calibrated to σ^0. Both filters were applied with a 7 x 7 pixels window and the Gamma MAP's look parameter L was set to 3 according to the ERS-2 image's specifications. The visual inspection showed that in general the Gamma MAP was able to preserve contrasts between homogeneous areas much better than the median filter, as expected. On the other side, the Gamma MAP was clearly outperformed in the computation speed by the median filter. Additionally, in the case of bright objects the Gamma MAP created a noise-like circle around them that. The artifact around a strong scatterer that dominates the local statistic is typical for filters that utilise local statistics. To conclude the filter review, because the main importance for us was the result of de-speckling not the computation time, the Gamma MAP method was selected for the system's de-speckling filter.

Having a de-speckled SAR scene on hand, the next step was to build the watermask by thresholding. A typical way to assign the thresholding cut-off value is to find the local minimum in the image's histogram. In the SAR image's histogram dark pixels representing open water areas are located on the left and land pixels to the right hand side . Between the distributions of water and land pixels resides the local minimum that represents the Bayes-optimal cut-off point for which the total classification error is lowest [36]. A weak spot in the method is that in the SAR image the distributions of land and water pixels typically overlap noticeably, even so badly that the local minimum does not exist. One important factor affecting to the existence of the local minimum is the ratio of land and water pixels in the image. If the percentage of water pixels is low compared to the percentage of land pixels, the local minimum most often does not exist. The other factor affecting to the separation of these distributions is the polarisation of the SAR image. Based on a set of ERS-2 (VV polarisation) scenes and a Radarsat (HH polarisation) scene (see Table 4.4), it was observed that a HH-polarised image had more distinctive separation between the distributions of land and water pixels than a VV-polarised image. Some of the ERS-2 images containing low number of water pixels or very disturbed water bodies manifested merged histograms of land and water so that it was impossible to find the local minimum between them. In the end the cut-off value was assigned empirically separately for the ERS-2 images and for the Radarsat image.

The watermask produced by thresholding needed further to be vectorised in order to use it for the initialisation of snakes. Because MATLAB's Image Processing Toolbox does not have built-in functions for vectorisation, the options were to code such a function, use an open source MATLAB function or use an external program. The programming of an own implementation was bypassed because of the estimated required time for the developing and testing of the vectorisation function would have been excessive. Next option was to search the Web for already available MATLAB functions for the vectorisation and test them. Having tried a variety of the vectorisation functions, it was quickly found out that they either had poor performance or were too slow for our purposes. Thus, the last hope was to search for an external program for the purpose. Commercial programs, based on their description, typically had complete graphical user-interface or application interfaces (APIs) that would have been hard to embed to MATLAB programming language. Finally, a simple freeware program called *ras2vec*[7] was found. The program was fast and accurate enough plus it run from the command-line making it easy to embed it to our system. Before inputting the water mask to the ras2vec program, it was cleaned from one-pixel areas and the borders of water areas were extracted by using MATLAB's morphological operations. Subsequently, having executed ras2vec, the result contours from the output text file were

[7]The program's web pages can be found at `http://xmailserver.org/davide.html`.

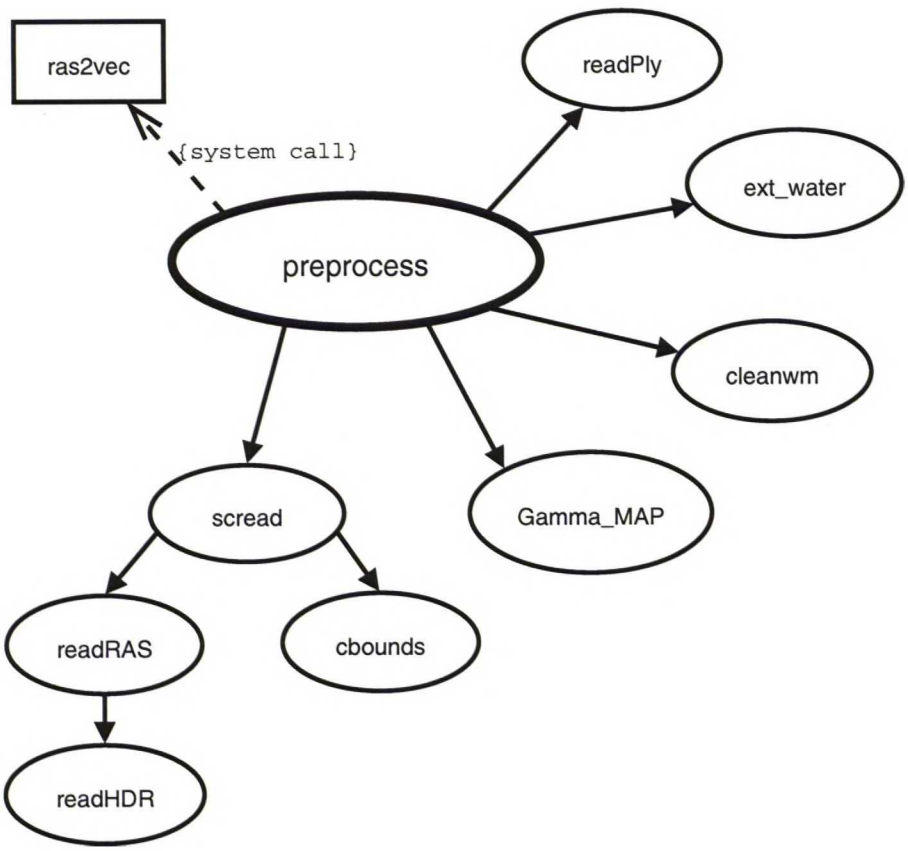

Figure 3.5: The dependency tree of the MATLAB functions implementing the SW development's second phase. A line with a single arrow indicates a one-way dependency.

read back to MATLAB.

The aforementioned phases comprise the automatic pre-processing step (see Figure 3.1) that was the goal of the second development phase. In function level the step is carried out by `preprocess` whose dependency tree is illustrated in Figure 3.5. In essence, the SAR scene is first de-speckled by `Gamma_MAP` whose output image is then thresholded by `ext_water`. The result water mask is subsequently cleaned from spurious one-pixel points by `cleanwm` and vectorised by using the ras2vec program. After couple intermediate phases the contours produced by ras2vec are ready to be used by the `initboa` function. The data flow from `preprocess` to `initboa` is illustrated in Figure 3.6.

3.3. Delineation of multiple flood areas

The third SW development phase's goal was to implement a parallel delineation of areas, i.e. to use multiple snakes at the same time. The dependency tree of `mainboa` for a single

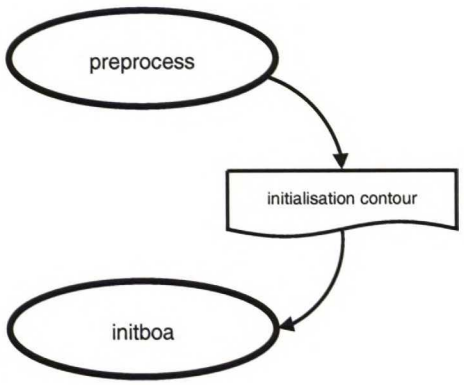

Figure 3.6: The data flow diagram from the preprocessing function to the function initialising snakes. The box with a curved lower border means data that are saved to a file.

snake (see Figure 3.3) grows up to the tree illustrated in Figure 3.7 when the functions needed for multiple snake processing are added. The new dependency tree's functions will be covered shortly as the corresponding issues in the delineation of multiple flood areas are explained.

The most challenging issue turned out to be to initialise a snake over a complex water body having islands that, in turn may contain water bodies, etc. For example, if a snake is initialised over a lake that has one or more islands, the seed statistics will be skewed by the pixels of the islands. Hence, we must exclude the islands' pixels when computing the seed statistics. Generalising the example leads to the fact that the system must be aware which water body includes which pieces of land and so on. Only when all relations, or the *topology*, are known it is possible to compute correct seed statistics for each water body or in the end for each snake.

The first problem was that the vector format water mask output by the ras2vec program did not provide any kind of information about the topology. Secondly, MATLAB had not built-in tree data type, which is normally used for describing hierarchical objects. The latter problem was sidestepped by using MATLAB's list data type for constructing a tree, whereas the former one required developing a method for the topology generation. A function `buildtopo` (see Figure 3.8) was written to process a vector format water mask and to convert it to a tree whose every *leaf* stores the contour of a water body or an island and contains *links* to the leafs on a lower level. The leafs linked to the *root* of a tree are on the *first level* and represent water bodies. The leafs connected to aforementioned leafs are on the *second level* and consequently represent islands and so on.

The `buildtopo` function starts by taking the first contour, say $\mathbf{u}_i$, and then finds if any contour $\mathbf{u}_k$ has any nodes inside $\mathbf{u}_i$. When the condition is found to match the

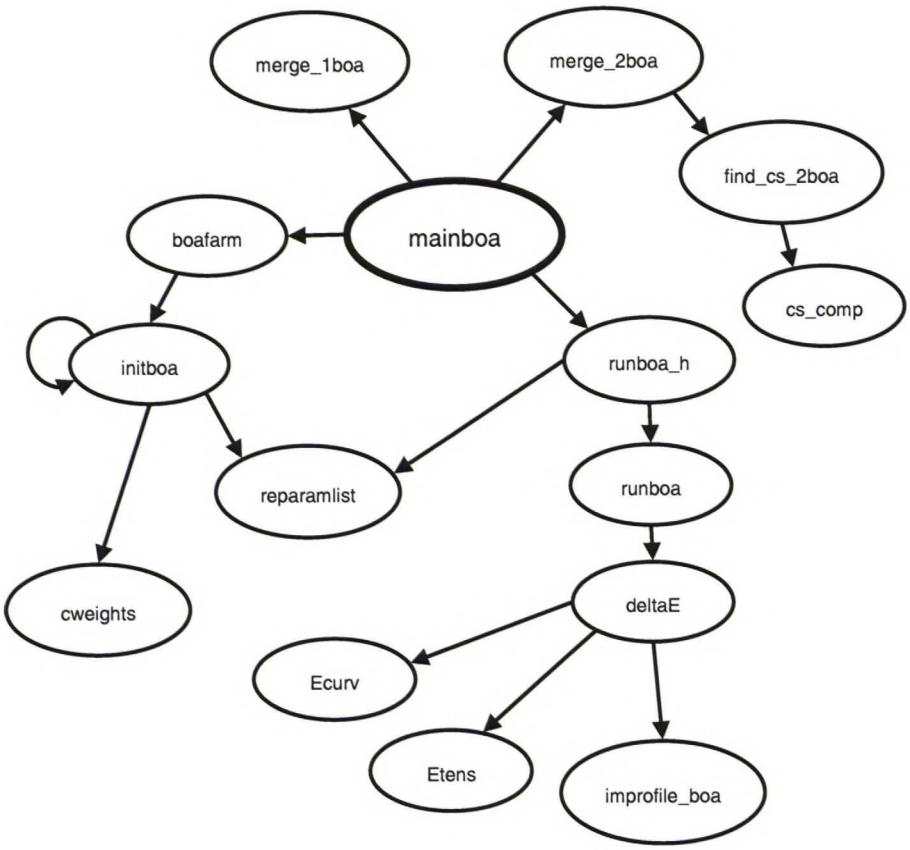

Figure 3.7: The dependency tree of the MATLAB functions implementing the SW development's first phase. A line with a single arrow indicates a one-way dependency and a line starting from and ending to the same function stands for a recursive call.

`find_cs_2boa` function is called to compute the possible intersections of the two contours. If no intersections are found, the obvious conclusion is that $\mathbf{u}_k$ is inside $\mathbf{u}_i$ and the creation of a *branch* to the tree is started by calling the `mktree` function[8]. Subsequently the process will continue similarly but now recursively by the function `robjects` that calls itself until no objects are found inside the water body or the island. In that case the control returns to `buildtopo` which takes the next contour $\mathbf{u}_{i+1}$ to processing. Thus, the building of the topology requires many nested loops that can be extremely time consuming, especially in the case of a big water mask containing a multitude of small areas.

Having built the topology tree, it is input to the `boafarm` function that is shown in Figure 3.7. The `boafarm` handles the computation of seed statistics taking into account

[8]Otherwise the contours are concluded to describe the different parts of a water body or an island and will be merged later, when the delineation process starts (see Step 2 in Figure 3.1). The merging will be shortly explained.

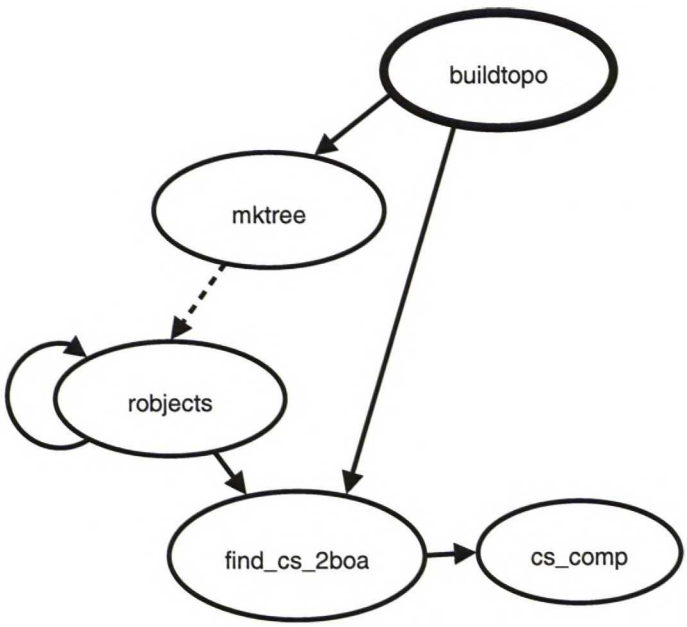

Figure 3.8: The dependency tree of the buildtopo function that creates the topology information. A line with a single arrow indicates a one-way dependency. Moreover, a line starting from and ending to the same function stands for a recursive call. A dash line, in turn, stands for a subfunction call.

possibly nested areas according to the topology tree. More exactly, the `initboa` function is called for initialising each first level water body. The `initboa` recursively calls itself until the whole branch has been processed and all water bodies and islands have been "converted" to snakes.

The process of running all snakes after the initialisation is fundamentally parallel, because each snake can operate independently. In a single CPU desktop machine the parallel processing is simulated by running each snake for a certain amount of iterations and then proceeding to the next one. It is clear that in a multiprocessor machine the algorithm could have a performance gain directly proportional to the number of processors. However, unlike in the C programming language, implementing the support for multiprocessor machines is not possible with the native MATLAB programming language. On the other side, the parallelism could have been added to the C version of the system's source code, as the conversion was needed in any case. Anyway, due to the size of the program, the required time for implementing the parallelism would have been excessive and thus the option was skipped. Instead, the processing was divided into three hierarchies illustrated in Figure 3.9. The hierarchies are represented by the nested boxes which are labelled according to a function that operates on the corresponding level. The biggest box, which is

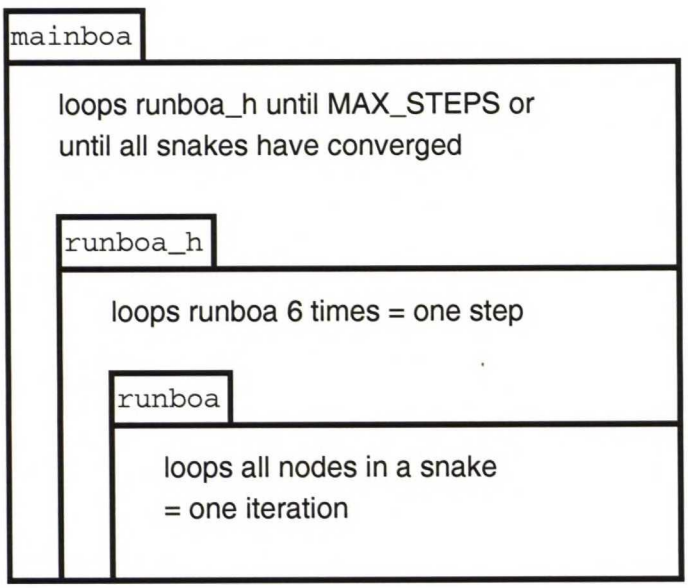

Figure 3.9: Three iteration hierarchies illustrated by nested boxes. A box's label refers to the function that carries out the computation on the corresponding hierarchy level.

labelled `mainboa`, corresponds to the highest processing level that means iterating through all snakes, one by one. The next level is executed by the `runboa_h` function and forms a processing unit called *step*. More exactly, a snake advances one *step* every time `runboa_h` is executed. A typical setting for one step is to loop `runboa` in total six[9]times. As the `runboa` function computes new positions for the snake's all nodes, a node can advance maximum 6 pixels during one step. As mentioned, in the end a snake is computed a new position by `runboa` that carries out the lowest processing level.

When running snakes successively they can at some point of time overlap or a single snake can develop a spurious loop so that the snake resembles the number "8", for example. In practise the former phenomena usually happens when the initialisation step has not properly found a water body, because of vegetation induced disturbances, for example. Thus, two or more snakes initialised over non-overlapping parts of the water body tend to grow together. The remedy for the situation is to merge these snakes to a new snake that covers their total area. As the seed statistics of the merged snakes may differ slightly, the new snake must be initialised with the `initboa` that computes new seed statistics from the new area. The merging itself is a trivial and computationally lightweight task, unlike the searching of each pair of snakes that need to be merged. Indeed, to find the intersecting snakes is an operation of $O((nm)^2)$ complexity, where n is the total number of snakes and

[9]The value provided a good compromise between the computation time spent to deforme, re-parametrise and merge snakes.

m is the average number of segments in a snake. Because of the computational burden, the merging by the `merge_2boa` function (shown in Figure 3.7) is run only after every 6th iteration.

As mentioned in the beginning of the previous paragraph, a snake settling to the target border can develop an erroneous extra loop. The phenomena can occur, for example, due to a difficult or a noisy border. To correct the deformation the extra loop is removed by the `merge_1boa` function shown in Figure 3.7. The operation is much less computationally intensive than the merging of two snakes, having a complexity of $O(nm^2)$, where n and m represent same parameters than in the `merge_2boa` function's case. The `merge_1boa` is run with similar scheduling than the `merge_2boa`, to allocate more computation time for the running of snakes.

3.4. Post-processing

The last phase of the SW development process was to solve the problem how to convert the converged snakes to such a data structure that they can be easily read to a remote sensing application. As each snake is represented by a polygon, the resulting water mask is naturally in a vector form. Thus, it would be convenient to be able to overlay the water mask on a SAR scene in a remote sensing application. A file format called *Shapefile*[10] is widely supported by remote sensing applications making it a respectable candidate for the purpose. Its drawback, however, is that MATLAB does not provide any tools for reading or writing Shape files. Neither any freeware, user-written MATLAB functions for the purpose were found from the Internet. As a matter of fact, even if another vector format designed to store GIS data had been selected, the support in MATLAB would have been as poor as with the Shapefile format. Having briefly browsed through Shapefile's file format specifications [37], a decision was made to implement the necessary functions from scratch.

Since only the reading and writing of polygons was needed, the programming task was easier than implementing the complete file format specifications which included several different shapes. The initial plan was to code `readSHP` and `writeSHP` for the reading and writing a **.shp* file that according to the Shapefile specifications contains the shape data or in this case the polygons. Having done this it was noted that some remote sensing applications neglected to read the files generated by `writeSHP`. In fact, Shapefile's specification list *three* necessary files for each shape data set. In addition to the aforementioned file, the two other files for indexing polygons and for providing text data field for each polygon

[10]The convention is that the type is referred as *Shapefile* but an instance of that type (or the actual file that is stored to the harddisk) is referred as *Shape file*.

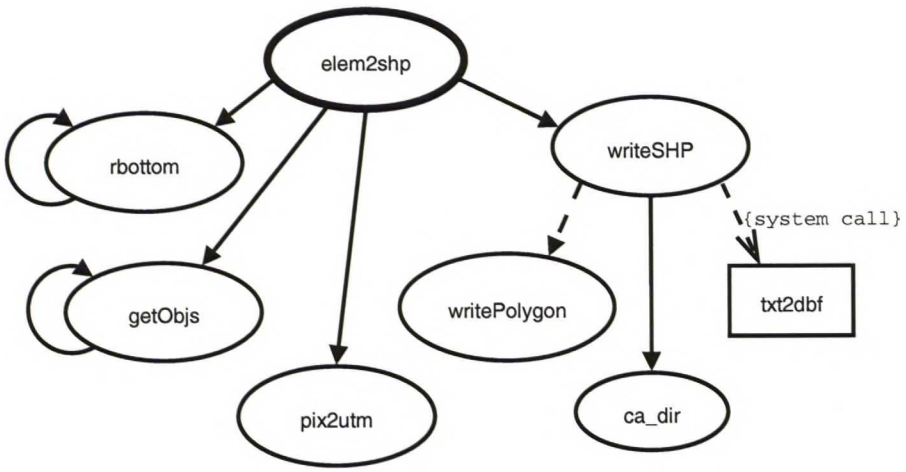

Figure 3.10: The dependency tree of the `elem2shp` function that converts the hierarchical water mask to a Shape file. A line with a single arrow indicates a one-way dependency. Moreover, a line starting from and ending to the same function stands for a recursive call. A dash line, in turn, stands for a subfunction call.

are needed. The former was easy to code with MATLAB but the latter, being a *dBASE*[11] format file, impossible because MATLAB once again lacked the I/O support for it. The solution was to use a freeware program called *txt2dbf*[12] that converts a formatted text file to a dBASE file. More exactly, the `writeSHP` function creates a text file containing running indexes for each polygon and then calls the txt2dbf program. Once txt2dbf has performed the conversion from the text to the dBASE, `writeSHP` deletes the dummy text file from the harddisk. The system call along with the `writePolygon` subfunction that are two main parts of `writeSHP` are illustrated on the right hand side in Figure 3.10. `WriteSHP` takes a structured list of which each element represents a water body along with its islands and loops it by calling `writePolygon` to write an element to the Shape file. To conclude, a lesson learnt in developing `readSHP` and `writeSHP` was to carefully study the file format specifications before heading to code functions that implement it.

As already mentioned `writeSHP` must be provided with a structured list of water bodies with their islands *before* the list can be written to a Shape file. In the pre-processing step the `buildtopo` function was used to build such a topology between water bodies and islands to a tree data structure. Hence, all that is needed to be able to store, for example, the initialisation contours to a Shape file for comparison with the result contours, is to implement a function that takes a tree data structure as an argument, converts it to a

[11] More information on dBASE format is available at *dataBased Intelligence, Inc.*'s web pages at `http://www.dbase.com`.

[12] The program's web pages are located at `http://www.vitsoft.info/dbf2txt.htm`.

structured list and finally passes the list to the `writeSHP` function. The described task is carried out by the `elem2shp` function, whose dependency tree is illustrated in Figure 3.10. As regards to storing of the resulting contours after the delineation step, the procedure follows the same principles than with the initialisation contours. The point that is worth of noting is that one could easily expect that the building of topology could be skipped here, because it already has been generated during the initialisation step. In reality the topology of water areas and islands changes during the delineation step because snakes may overlap causing the need of merging, etc. In addition a snake whose size is close to the minimum (4 nodes) may start to shrink, though it is expected to grow. Further, when the re-parametrisation is carried out for the snake, its node count drops under the minimum and it is erased. The depicted situation may happen if areas that do not possess similar statistics than water in the SAR image have been detected as water by the thresholding and subsequently used to for the initialisation of some snakes. The reasons for mis-detection of land and water were discussed earlier in the second development phase. Finally, the merging and the erasing result that after all snakes have converged the topology must be built once again.

The topology generation in both the pre-processing step and the delineation step are visualised in Figure 3.11 that embeds the dependency tree (on the left hand side) and the data flow diagram (on the right hand side) of the complete system after the fourth development phase. As illustrated in Figure 3.11, the pre-processing and the delineation steps are respectively carried out by the `AC_Stage1` and the `AC_Stage2` function.

In the very beginning of this chapter the goal was set to design a stand-alone application for the flood delineation. So far we have completed the development of the two step program that, however, still runs only in MATLAB. Hence, the final task was to convert the M-code to a stand-alone application by using MATLAB's built-in compiler. Though the conversion at first sight seemed to be trivial, it was soon found out that the compiler does not support all similar programming conventions than MATLAB's interpreter that runs the M-code. The result was that fundamental structure of the whole system consisting of about 25 functions of which the largest was hundreds of lines long had to be modified to comply with the compiler's requirements. Having completed the modifications there were no severe problems to compile the M-code to a stand-alone application. Finally, after the debugging and the testing of the stand-alone application, the SW development process was completed.

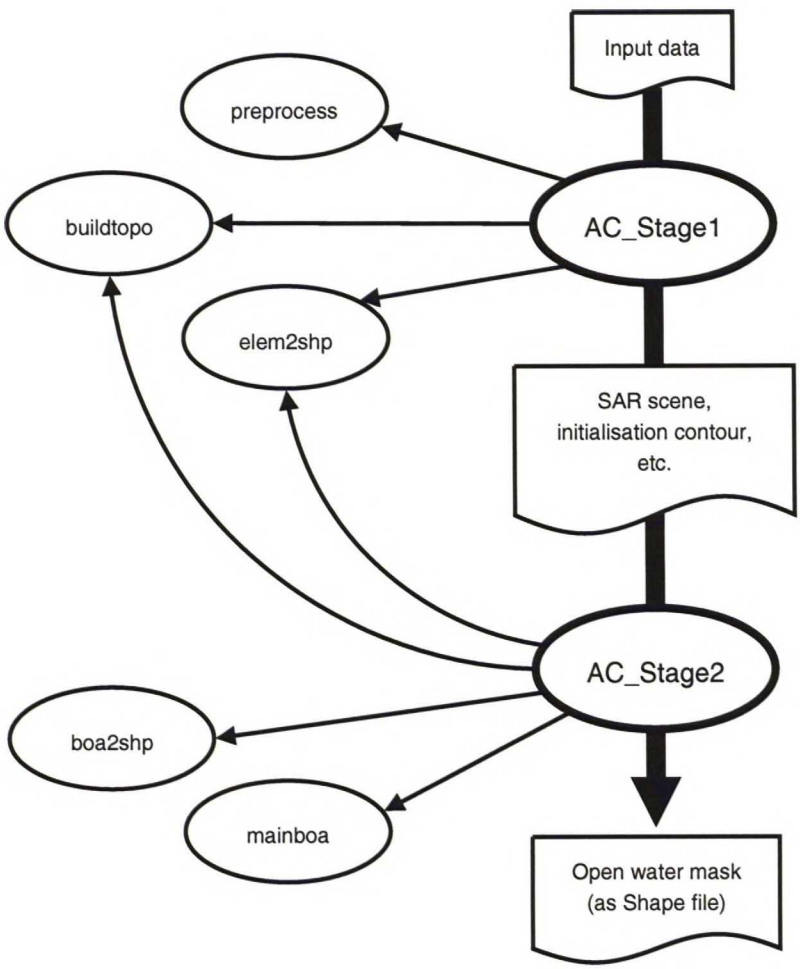

Figure 3.11: The data flow diagram combined with the dependency tree of the system's two stages. The bold arrow indicates the data flow and the smaller arrows the dependencies between functions. `AC_Stage1` and `AC_Stage2` are the preprocessing and the delineation stages that were initially illustrated in Figure 3.1.

Chapter 4

Evaluation of the flood delineation system

The goal of the evaluation was both qualitatively and quantitatively test the overall performance of the designed system. The common convention for testing a classifier is to establish a grid of test points on the test area and then classify these points. The overall picture of the classification performance that aforementioned kind of test configuration can give is strongly correlated to the number and spacing of the test points. If the number of test points is relatively high compared to the size of the test area and they are uniformly distributed over it, the results are much more comprehensive than the results derived by using only a bunch of test points. Unfortunately, usually the ratio of test points and total amount of pixels is very low. For example, as mentioned in Section 2.2, in [15] and [17] respectively 11 and 202 test points for a 100 x 100 km, 30 m resolution Radarsat scene were used. In these cases for the aforementioned ratio we get $9.9 * 10^{-7}$ and $1.8 * 10^{-5}$. Obviously, these results in [15] and [17] reporting over 90 % correct classification rates can be strongly questioned. However, it is worth of noting that especially in the case of testing the flood mapping performance of a classifier it is hard to obtain reference data depicting the situation at the moment of the SAR satellite pass. Moreover, considering the floods in Finland, a typical spring flood rises and fades in a few days that makes it indeed crucial to get SAR and reference data whose acquisition times are within hours.

The rest of this chapter is organised so that 4.1 describes the data that were used in the qualitative test cases that are presented under 4.2. Finally, the quantitative testing along with its data are explained in 4.3.

Table 4.1: SAR scenes that were used for qualitative testing. Each scene (100 x 100 km) was geocoded to the UTM-35 map projection in the WGS-84 coordinate system yielding a 25 m resolution. In the *Pass* column, *dsc* and *asc* stand for descending and ascending satellite passes, respectively.

Scene area	Date	Type	Orbit-frame	Pass	Flood events
Kyrönjoki	20000422	ERS-2 PRI	26171-2331	dsc	Iskala, Lapuanjoki
Kyrönjoki	20000425	ERS-2 PRI	26214-2331	dsc	Iskala, Lapuanjoki
Kyrönjoki	20000527	ERS-2 PRI	26672-2331	dsc	-
Kyrönjoki	20000530	ERS-2 PRI	26715-2331	dsc	-
Kyrönjoki	20000805	ERS-2 PRI	27674-2331	dsc	-
Kyrönjoki	20010410	ERS-2 PRI	31224-2331	dsc	Lapuanjoki, Rintala
Kyrönjoki	20010724	ERS-2 PRI	32727-2331	dsc	-
Kisko	20040805	Radarsat-SGF S2	not available	asc	-

4.1. Test setup

For the qualitative testing we obtained seven ERS-2 scenes (see Table 4.1) between 22nd April 2000 and 24th July 2001 for River Kyrönjoki region in Western Finland. Because of the plains surrounding Kyrönjoki's riverbed, the area is very prone to spring floods during the snow melting period. Indeed, three of these scenes capture spring flood events, as listed in Table 4.1. In addition to the ERS-2 scenes, one Radarsat scene that was primarily acquired for the quantitative testing was as well used for the qualitative tests. The Radarsat scene was acquired on 5th August 2004 for Kisko region in Southern Finland. The region contains mainly agricultural and forested flat plains with multitudes of small and medium sized lakes.

The specifications of both the ERS-2 scenes and the Radarsat scene are presented in Table 4.2. Before the scenes were input to our system, they were geocoded to the UTM-35 map projection with the WGS-84 coordinate system by using a DEM having a 25 x 25 m grid. Because of the grid size of the DEM the produced geocoded images were bounded to the 25 m resolution. During the geocoding step the ERS-2 scenes were re-sampled from 12.5 x 12.5 m pixel size to the DEM's pixel size that was 25 x 25 m, as noted.

To get an overall picture of the system capability to separate land and water areas we initially planned to acquire a SAR scene and a high resolution multispectral image for a flood event in Finland. Further, it was decided that the acquisition times should be as

	ERS-2 PRI	Radarsat-SGF S2
Incidence angle	23°	27.5°
Band and polarisation	C-band, VV pol.	C-band, HH pol.
Scene's nominal size	100 x 100 km	100 x 100 km
Pixel size	12.5 x 12.5 m	25 x 25 m
Number of looks	3	3
Projection	ground range	ground range

Table 4.2: Specifications of used ERS-2 and Radarsat image products.

close to each other as possible, like stressed in the beginning of this chapter.

4.2. Test cases

The focus in the first tests was to visually observe the system's delineation performance by using the ERS-2 scenes listed in Table 4.1. The scenes capturing a flood event or flood events were given especial attention by studying them with all available ground truth information. Indeed, the ground truth information was available only for two flood events that will be covered in the following. These tests comply with our plan to first test the delineation performance qualitatively.

The spring flood cases on 20th April 2000 in Iskala and on 11th April 2001 in Rintala had ground truth data about the water level during the flood and were thus selected for the evaluation. Both regions are located to Western Finland whose rivers and vast flat plains make the area prone to spring floods, e.g. due to the melting of snow or due to the hanging ice dams forming to rivers. In Table 4.1 it can be seen that none of the ERS-2 scenes' acquisition times matches exactly with aforementioned flood events. The flood boundaries at the time of the ERS-2 overpasses on 22nd April 2000 and at on 10th April 2001 were approximated by using the recorded water level data and a fine resolution DEM. These flood boundaries were generated by Finnish Environment Institute.

4.2.1. Iskala flood

In Iskala flood event the water level data was recorded by a pump station whose measurement gauge was adjoined to the inundated area. Unfortunately, because of the station operator's error the water level measurements during the flood had been slightly corrupted by the pump turned on just on the side of the measurement gauge. Consequently, because

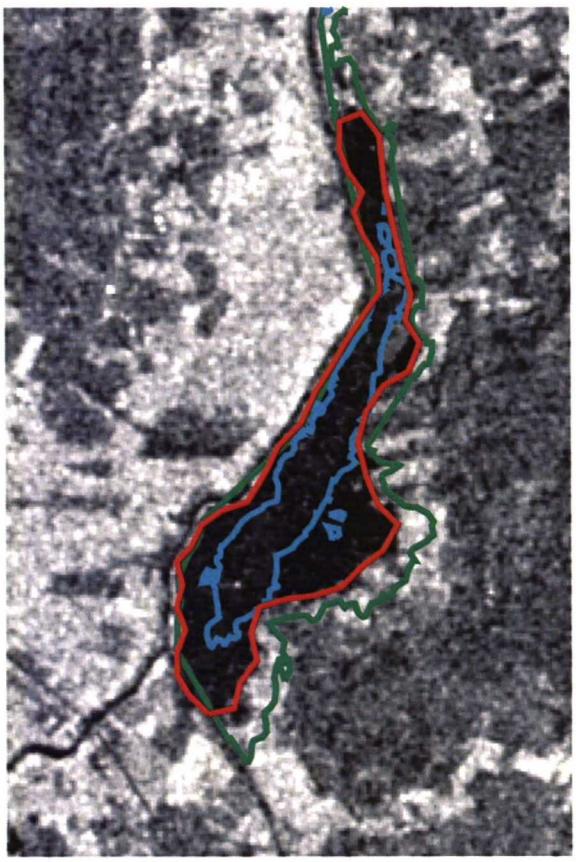

Figure 4.1: A 5 x 8 km crop of the ERS-2 scene capturing the Iskala flood event at 12:40 (UTC+3) on 22nd April 2000. The red polygon represents the delineation result and the polylines coloured to cyan and green represent different approximations for the flood boundaries during the ERS-2 pass. More exactly, the cyan and the green contours correspond to 37.14 m and 38.00 m water level in Finnish N60 height system.

of the strong flow the water level measurements contain an unknown offset. In the end Finnish Environmental Institute generated approximations for the lowest and for the highest water level of the inundated area. The respective bounds are 37.14 m and 38.00 m, both in Finnish N60 height system[1]. The corresponding flood boundaries are illustrated in Figure 4.1 by the cyan and the green contour, respectively. The nearly 1 meter difference between the flood level approximations results a vast areal difference of approximated flood boundaries. Further, by visual inspection it is easy to note that the flood boundary derived from the lowest water level approximation does not match to the inundated area (dark pixels in the image) at all. The upper estimate provides a better match, though clearly

[1]The N60 system's zero level coincides with the mean water level in Helsinki in 1960. For more information about the N60 height system, look at `http://www.maanmittauslaitos.fi`.

overestimating the inundation area. However, not having an accurate reference polygon for the moment of the ERS-2 pass, we can only but speculate whether or not the inundated bank areas having some emergent vegetation are correctly delineated. In addition, on the bank areas the delineation result could depict the border much smoother than it does in Figure 4.1 on the preceding page, though the minimum limit for the link length between nodes is 5 pixels.

4.2.2. Rintala flood

The flood event on 11th April 2001 in the Rintala region occurred when, according to the local regulations, the doors of a flood wall were opened due to the rising water level in River Kyrönjoki. The wall protects villagers from a modest spring flood. In case of a bigger flood, water must be directed to open fields via the doors. Consequently, the inundated areas in Figure 4.2 can be seen on the both sides of the river, clearly separated from it.

During the scene's acquisition time the flood was rising as the water was flowing to the fields. Thus, the surface seen by ERS-2 was partly covered by water and by land spots. Indeed, in Figure 4.2 the flood plains have only some black spots acting like specular scatterers. Moreover, in some places the flood plain's texture resembles closely the texture of forested areas (lower right on the image). When we input the image to the system's pre-processing, it found exactly the aforementioned spots. As the black areas did not exhibit the similar statistical variation than the other parts of the flood plains, it was expectable that the system's delineation step was not able to expand the Active Contours from the initialisation areas. However, if the Active Contours were initialised manually the result was much better. Figures 4.2 on the next page and 4.3 on page 54, both containing the same red result contour, show that when initialised properly the Active Contour is able to find and delineate an area that contains desired type of texture and possesses desired statistics.

The delineation result after the manual initialisation was compared to the southern flood boundary (see Figure 4.3 on page 54) that was digitised from geocoded aerial photos that have been taken around at 12:00 (UTC+3) on 11th April 2001. For the northern flood plain no aerial photos were available. In any case, the fact that the flood was rising during the ERS-2 pass and reached its peak on the following day when the aerial photos were taken made the comparison impossible. Indeed, it can be seen in Figure 4.3 on page 54 that the flood plain on 11th April is much larger than it had been a day before.

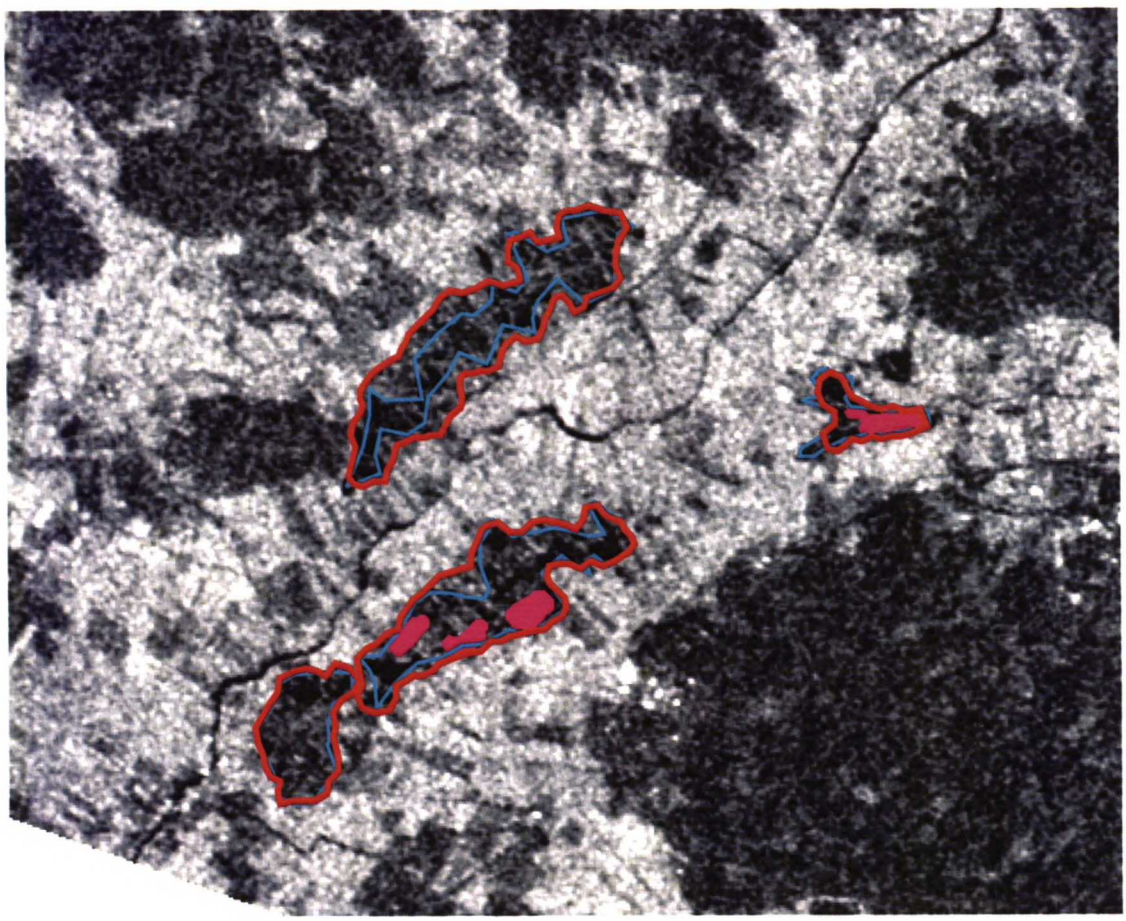

Figure 4.2: An 11 x 9 km crop of the ERS-2 scene capturing the Rintala flood event at 12:45 (UTC+3) on 10th April 2001. The areas filled by magenta colour represent the result obtained by the automatic initialisation. The red polygons, in turn, represent the algorithm's result obtained by the manual initialisation with the cyan polygons.

4.3. Accuracy validation

Having performed the qualitative tests according to the plan, the quantitative part was to validate the accuracy of the delineation system, as mentioned in Section 4.1.

The timespan for the acquisition of a SAR and an airborne image of a flood event was from the end of March to the beginning of June in 2004, or in other words, the late spring in 2004. Unfortunately, during Spring 2004 did not occur any flood that would have been large enough to be seen even on an 8 m resolution[2]. Thus, the original plan had to be changed so that a SAR scene for a water body whose extent corresponds to approximately

[2]The 8 m resolution is the finest that can be obtained from a commercial SAR satellite, in this case by using Radarsat's *Fine Beam Mode*.

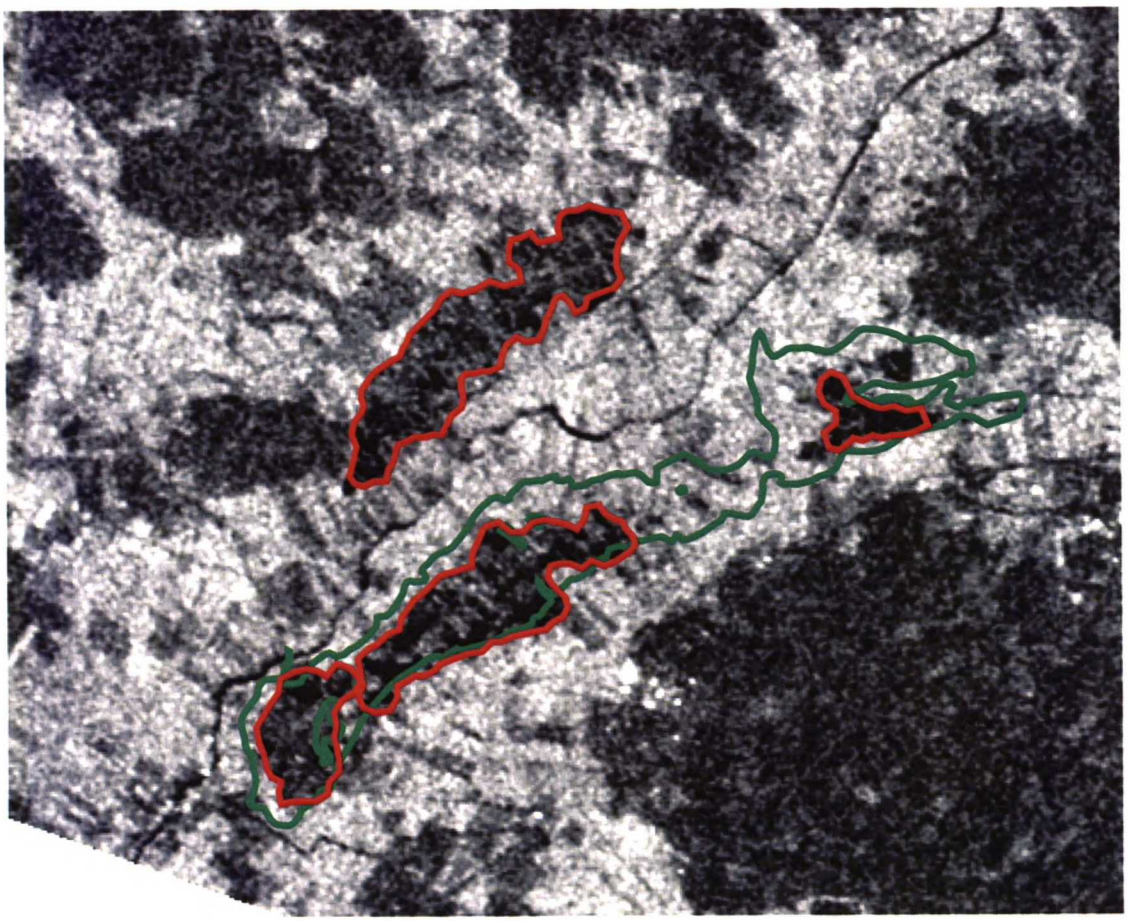

Figure 4.3: An 11 x 9 km crop of the ERS-2 scene capturing the Rintala flood event at 12:45 (UTC+3) on 10th April 2001. The red polygons represent the delineation result. The green line depicts the southern flood plain that has been digitised from aerial photos that have been acquired ca. 24 hours later that the ERS-2 scene.

to the extent of an average spring flood in Finland would be ordered. Additionally, a flight campaign to acquire a high resolution multispectral image for the water body would be carried out during the same day than the SAR satellite pass would occur.

4.3.1. Dataset

Satisfying the size requirement, Lake Kirkkojärvi (ca. 1.7 x 7.9 km) in Kisko region in Southern Finland was selected to be the test area for the quantitative testing, or in other words, for the accuracy validation. Proceeding with the new image acquisition plan we prepared to order a Radarsat scene and planned an AISA flight campaign (see Table 4.3) for the beginning of August 2004. The main constraining factor for the flight campaign was

AISA specifications	Value
Wavelength range	440-890 nm
Field of view (FOV)	21°
Swath width	720 m
Pixel size	2 x 2 m

Table 4.3: Specifications of the AISA airborne imaging spectrometer at the flight altitude of 2000 meters. The AISA instrument was mounted to Short Skyvan research aircraft and Lake Kirkkojärvi was imaged in total 4 parallel strips that were mosaicked and geocoded later to the UTM-35 map projection in the WGS-84 coordinate system.

Image area	Date	Type	Acquisition time
Kirkkojärvi	20040805	Radarsat-SGF S2	18:54
Kirkkojärvi	20040805	AISA spectrometer image	10:54-11:38

Table 4.4: Image data that were used for the accuracy validation. The Radarsat image (9 x 7 km) was cropped from the full size scene to cover Lake Kirkkojärvi and its nearby areas.

the weather forecast for next couple days, especially as we wanted cloud free conditions over Lake Kirkkojärvi. Another factor making the scheduling harder was Radarsat's 48h rush order time, meaning that the scene had to be ordered 2 days prior to the imaging. Finally, on 5th Autumn 2004 we carried out the flight campaign in clear sky conditions and obtained a multispectral image of Lake Kirkkojärvi. During the 6-hour offset in the acquisition times (see Table 4.4) of the multispectral image and the Radarsat scene the weather conditions remained stable. Consequently, it is reasonable to assume that the multispectral image's shorelines act as a valid reference for the moment when the Radarsat image presented in Figure 4.4 was acquired.

A misfortune that was encountered in the very beginning of the flight campaign was that the GPS system that records the location, velocity and acceleration vectors for the research aircraft malfunctioned after we had imaged only two strips covering the northern shoreline of Lake Kirkkojärvi. The data recorded by the GPS system is vital to be able to mosaic and geocode the recorded image data properly. Unfortunately the GPS data from the two southern passes over the lake turned out to be so deprecated that they were not usable for the geocoding purposes. Nevertheless, we managed to mosaic and geocode the area formed by the two first passes over the lake. In Figure 4.5 on page 57 the RGB

Figure 4.4: Lake Kirkkojärvi validation area in a 7 x 9 km crop of the Radarsat scene acquired on 5th August 2004.

composite AISA image mosaic of the two strips is overlaid on a high resolution digital map covering Lake Kirkkojärvi region. The mosaicked and geocoded image was manually moved by 70 meters to get a good match with the digital map.

Having the correctly geocoded the 13-km long northern shoreline image strip, it was vectorised by using AISA spectrometers near infrared (NIR) image yielding a reference polyline with 2 m accuracy. More exactly, Lake Kirkkojärvi's water extent area was first thresholded in the NIR image and then vectorised by following a similar procedure than in the pre-processing step that was described in Chapter 3. The only difference was that in this case the procedure was done manually to obtain the best possible thresholding result. Based on the visual comparisons between the vectorised northern shoreline and AISA's images of visible wavelengths, it was confirmed that the cumulative error in the thresholding and the vectorisation fell below 12 AISA pixels, or in other words, below one Radarsat pixel. In summary, the produced polyline of the northern shoreline can be considered as a valid *reference polyline* for the accuracy validation.

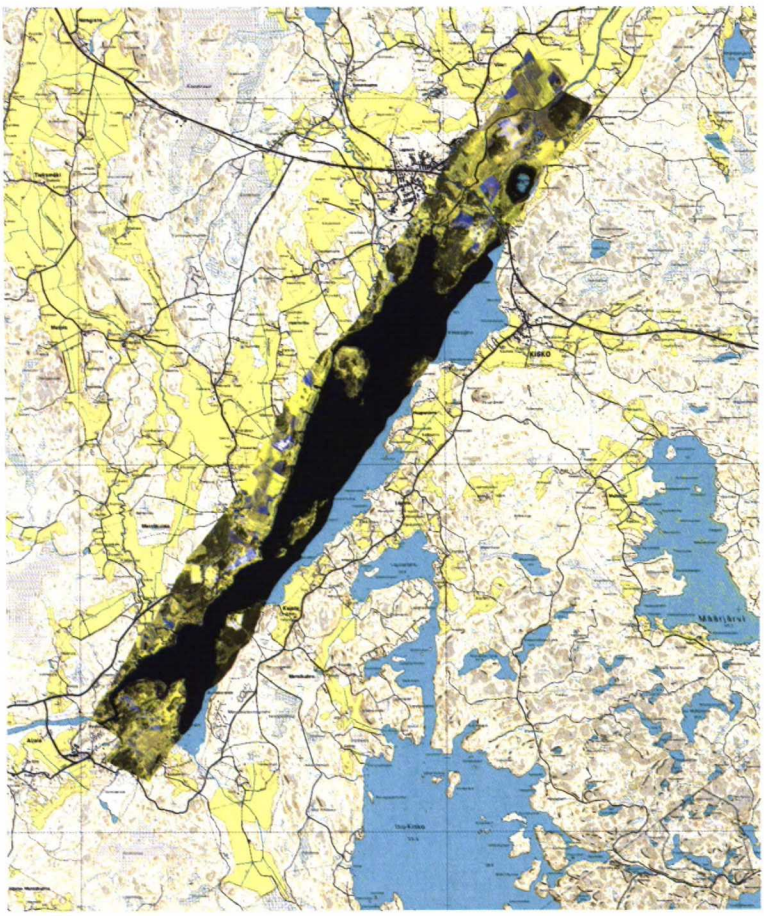

Figure 4.5: Two northern AISA image strips mosaicked and overlaid on the high resolution digital map covering an area of 10.2 x 12.3 km in Kisko region. Both the AISA image mosaic and the digital map are in Finnish KKJ2 map projection.

4.3.2. Error distance measure

To begin the accuracy validation we adopt a notation in which the reference polyline of Lake Kirkkojärvi's northern shoreline is marked by $\mathbf{u}_{ref}$ and the delineation system's result by $\mathbf{u}_{AC}$. Respective polylines having 511 and 82 line segments are represented by the blue and the red lines in Figure 4.6. Further, to measure the difference between $\mathbf{u}_{ref}$ and $\mathbf{u}_{AC}$, an *error distance measure* was created. The error distances were computed from $\mathbf{u}_{AC}$'s each line segment's mid-point to the nearest point on $\mathbf{u}_{ref}$ along the line segment's normal. Figure 4.7 on page 59 illustrates these error distances by 82 green line segments. Additionally, red and blue polylines illustrate $\mathbf{u}_{AC}$ and $\mathbf{u}_{ref}$, respectively.

The computed error distances were aligned to a histogram after which it was scaled in order to obtain a CDF that is represented in Figure 4.8 on page 59. The CDF shows

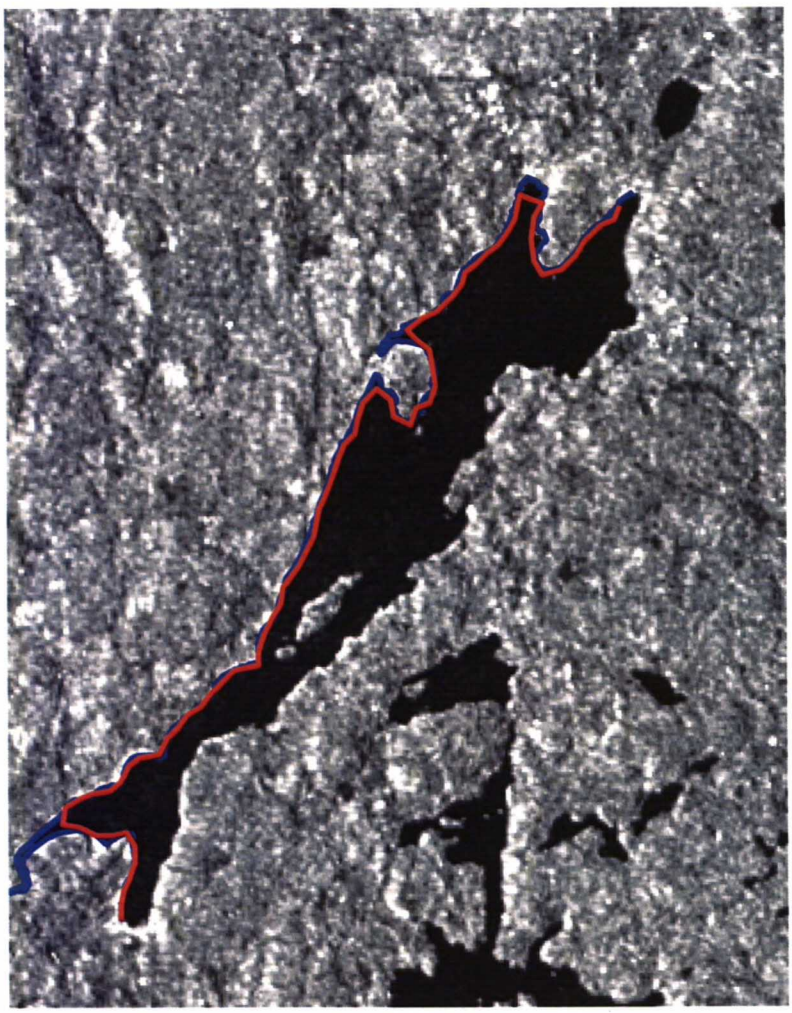

Figure 4.6: Lake Kirkkojärvi validation area in a 7 x 9 km crop of the Radarsat image acquired on 5th August 2004. The blue and red line correspond to the reference and the Active Contour's result, respectively. The result polygon was cut to begin and end in about same locations than the reference polyline.

that 70 % of the result lines segments are within 25 m and 95 % within 50 m from the AISA reference line. The corresponding distances in Radarsat pixels are 1 and 2 so that, for example, 95 % of the Active Contour's result is within 2 pixels of the reference line. Further, the minimum and the maximum error distances (see Table 4.5 on page 60) correspond to approximately 1 and 7 Radarsat pixels, respectively. The maximum error distance occurs in the lower left corner in Figure 4.6 where the AISA reference line is capable to delineate the mouth of a river much further to inland than the Active Contour's result line.

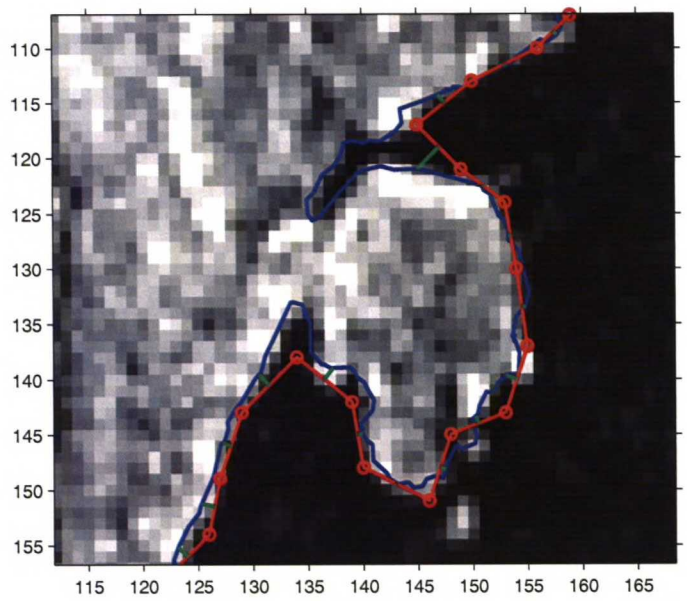

Figure 4.7: Zoomed 1.3 x 1.5 km portion of Lake Kirkkojärvi in Figure 4.6. The blue and red line correspond to the reference and the Active Contour's result, respectively, and the green lines represent the error distances.

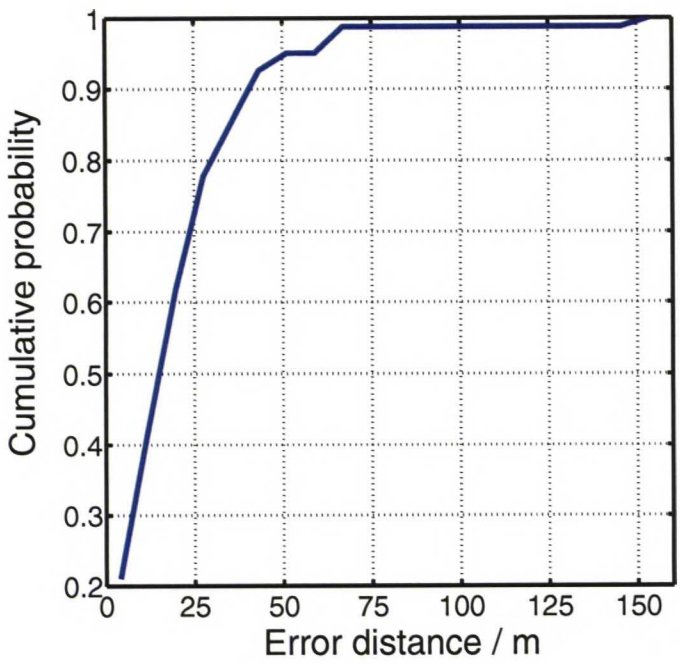

Figure 4.8: Cumulative distribution function of the error distances.

Statistic	Value (m)
Min	0.13
Max	157.4
Mean	22.7
Std	21.8

Table 4.5: Statistics of the computed error distances.

Chapter 5

Conclusions

The development of the delineation system for MATLAB that has no built-in support for remote sensing applications turned out to be very time consuming. In spite of the time consumed to tackle with the implementation issues, we managed to build a stand-alone system that can be run in a Linux or a PC workstation without problems. A drawback is that due to the processing steps and the conversions between raster and vector format, the system is considerable complex.

The Rintala flood event proved that the automatic initialisation by thresholding is by no means a robust method for the purpose. On other hand, the delineation result after the manual initialisation turned out to be superior compared to the thresholding's result. In that way the goal to be able to delineate inundated areas manifesting bright textures can be considered fulfilled.

In the Iskala event we had a flood that had already reached its peak, unlike in Rintala, resulting a smooth surfaced inundated area. Consequently, the automatic initialisation was able to pick a good estimate for the extent of the flood plain. The initialisation contour was so close to bank areas that, in fact, the Active Contour converged only after couple steps. In the end, by visual inspection the result polygon provided a good correspondence to the inundated area.

Turning the focus to the accuracy validation, most of the Active Contour's result line fell within two Radarsat pixels, as noted in Section 4.3.2. The error is comparable to the results that have been reported in earlier applications [12, 9] of the Active Contour. On the other hand, the validation scene did not represent a real flood event but rather a water body having clear borders and no emergent vegetation or any other kind of roughness. Would the accuracy be as good as in Lake Kirkkojärvi also in a rough inundated area like Rintala can be questioned until we can obtain a reference image and a SAR scene from such a flood event.

An important issue affecting the delineation accuracy is the parameter selection for

the Active Contour algorithm. Being a numerical algorithm it requires several internal constants that need to be set empirically. Further, as the statistics of an inundated area are strongly affected by its surface roughness, the manual parameter adjustment was needed in our tests to achieve best possible results. Considering more robust operation, an automatic tuning could be achieved for example by using a small neural network that incorporates a priori information on the statistics of inundated areas and the suitable internal constants for them.

In summary, the initially set goals were partly fulfilled, though there remains much work to do. In Rintala we were able to successfully delineate a rough inundated area but were not able to perform an automatic initialisation. Hence, the further work includes to develop a method to reliably initialise the Active Contour. Further, the revised system should be tested by similar manner than in this study.

Appendix

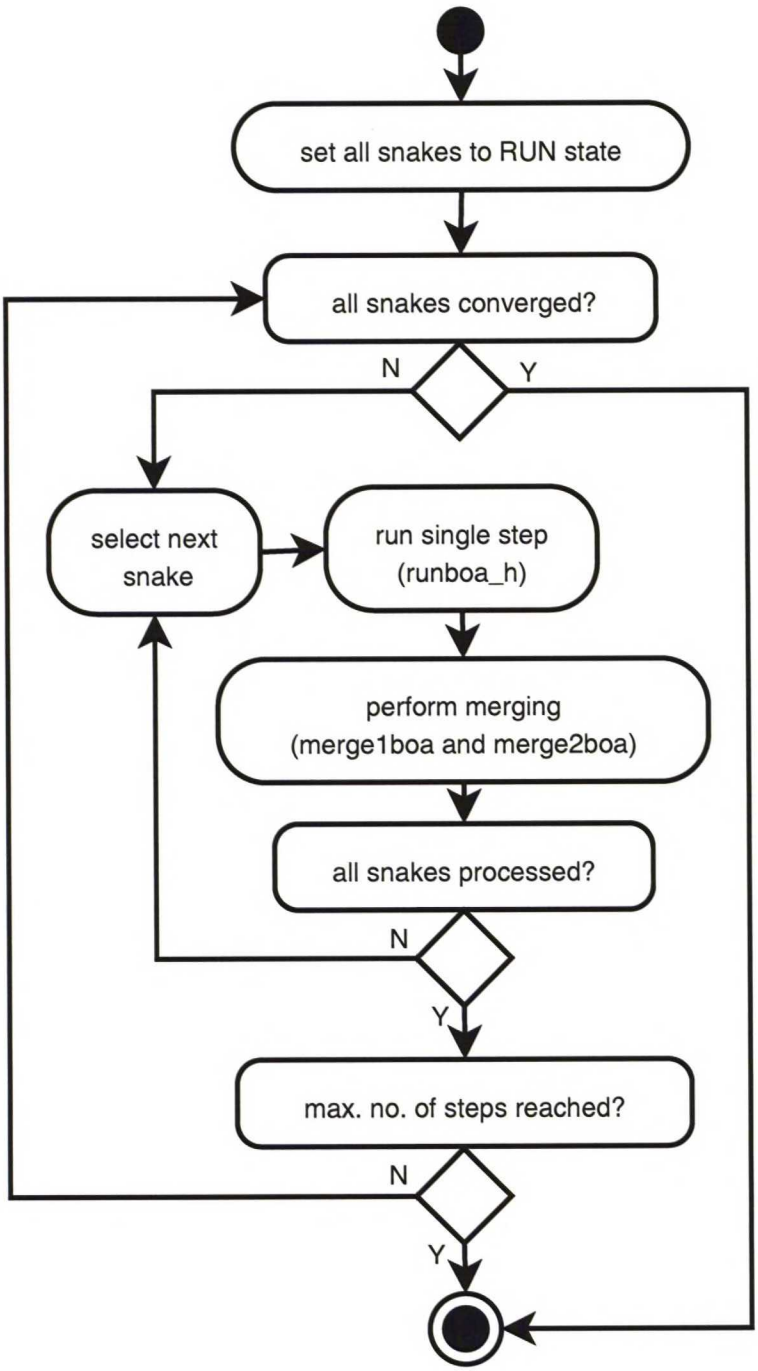

Figure 5.1: The state diagram of the `mainboa` function.

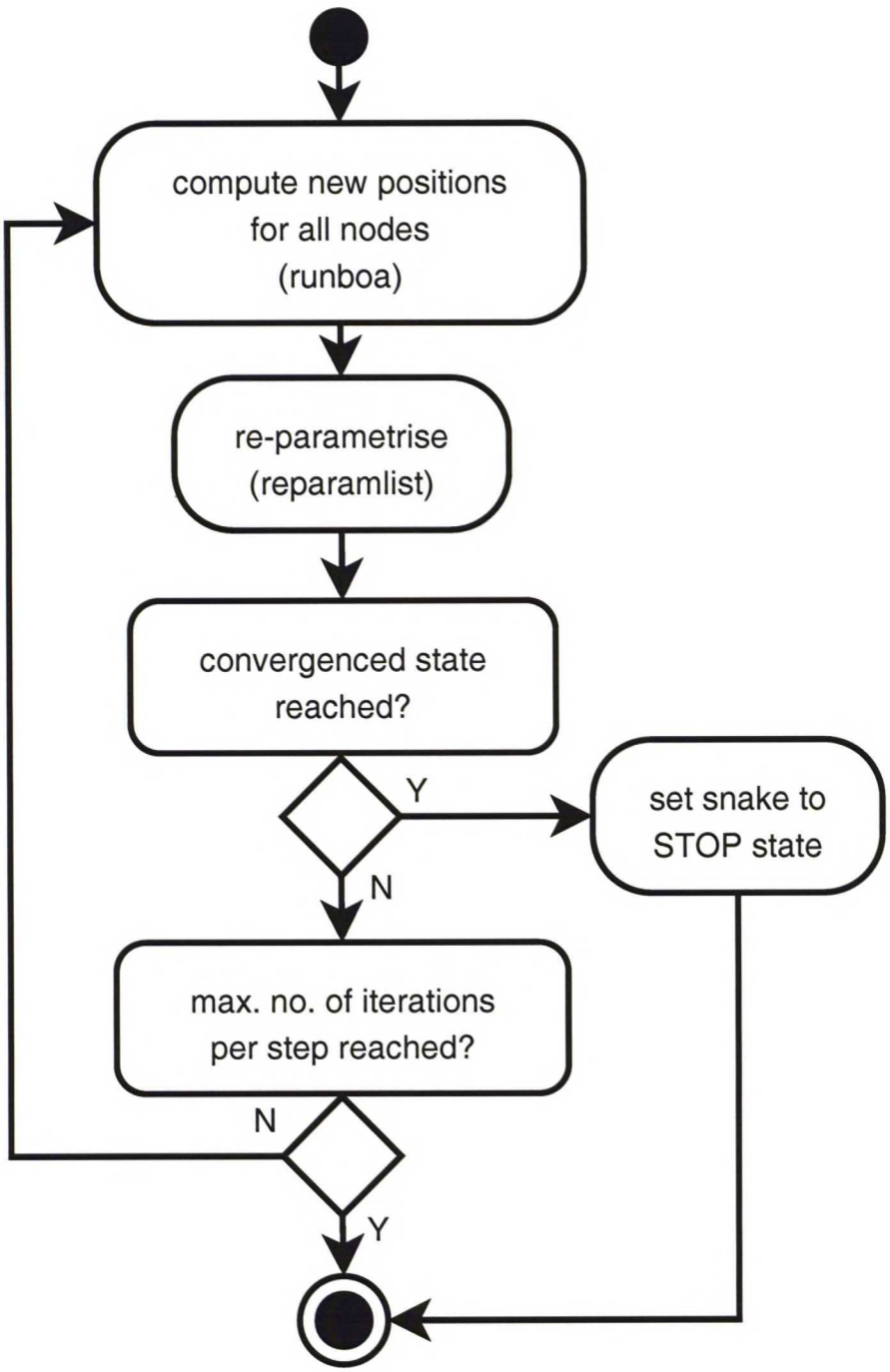

Figure 5.2: The state diagram of the `runboa_h` function.

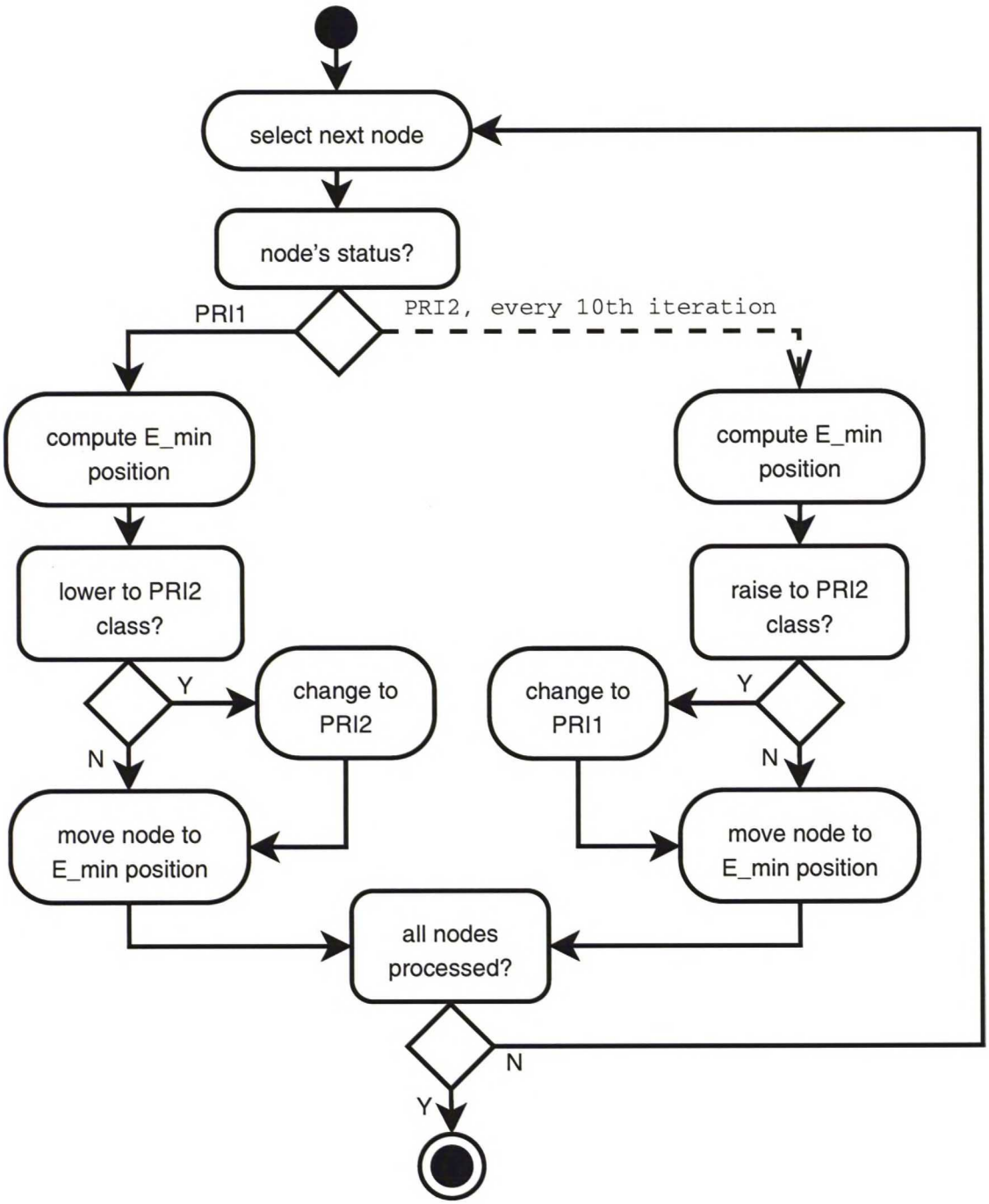

Figure 5.3: The state diagram of the `runboa` function. The diagram obeys the Unified Modelling Language (UML) notation except the slashed line that stands for conditional execution, depending on the iteration round.

Bibliography

[1] R. Corell and et. al. (2004, 14 Nov.) ACIA overview report. Arctic Council. [Online]. Available: http://www.amap.no/acia

[2] R. Timonen and et. al. (2003, 15 Mar.) Suurtulvatyöryhmän loppuraportti. [Online]. Available: http://www.ymparisto.fi/download.asp?contentid=23361&lan=FI

[3] C. Oliver and S. Quegan, *Understanding Synthetic Aperture Radar Images.* Norwood, MA: Artech House, 1998.

[4] C. Elachi, *Spaceborne Radar Remote Sensing: Applications and Techniques.* New York, NY: IEEE Press, 1987.

[5] A. Reigber. (2004, 15 Nov.) Synthetic aperture radar - basic concepts and image formation. Epsilon Nought - Radar Remote Sensing. [Online]. Available: http://epsilon.nought.de/tutorials/processing/index.php

[6] A. Moreira, "SAR remote sensing - principles and theory," 15–19 June 2003, HUT remote sensing summer school handouts.

[7] M. Hallikainen, "S-92.131 Remote Sensing," 2003, remote sensing course handouts.

[8] L. Hess, J. Melack, and D. Simonett, "Radar detection of flooding areas beneath the forest canopy: a review," *International Journal of Remote Sensing*, vol. 11, no. 5, pp. 1313–1325, 1990.

[9] M. Horritt, D. Mason, D. Cobby, I. Davenport, and P. Bates, "Waterline mapping in flooded vegetation from airborne SAR imagery," *Remote Sensing of Environment*, vol. 85, pp. 271–281, 2003.

[10] ESA Earthnet. (2004, 23 Nov.) 23. Parameters affecting radar backscatter. [Online]. Available: http://earth.esa.int/applications/data_util/SARDOCS/spaceborne/Radar_Courses/Radar_Course_II/parameters_affecting.htm

[11] ——. (2004, 23 Nov.) SAR design. [Online]. Available: http://earth.esa.int/rootcollection/eeo4.10075/eeo3.298.html

[12] M. Horritt, D. Mason, and A. Luckman, "Flood boundary delineation from synthetic aperture radar imagery using a statistical active contour model," *Remote Sensing of Environment*, vol. 22, no. 13, pp. 2489–2507, 2001.

[13] J. Lee and I. Jurkevich, "Coastline detection and tracing in SAR images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 28, no. 4, pp. 662–668, 1990.

[14] S. Solbo, E. Malnes, T. Guneriussen, I. Solheim, and T. Eltoft, "Mapping surface-water with Radarsat at arbitrary incidence angles," in *Proceedings of International Geoscience and Remote Sensing Symposium*, vol. 4. IEEE International, 21–25 July 2003, pp. 2517–2519.

[15] P. Townsend, "Mapping seasonal flooding in forested wetlands using multi-temporal Radarsat SAR," *Photogrammetric Engineering and Remote Sensing*, vol. 67, no. 7, pp. 857–864, 2001.

[16] M. Imhoff, C. Vermillion, M. Story, A. Cloudhury, A. Gafoor, and F. Polcyn, "Monsoon flood boundary delineation and damage assessment using space borne imaging radar and landsat data," *Photogrammetric Engineering and Remote Sensing*, vol. 53, no. 4, pp. 405–413, 1987.

[17] P. Townsend, "Relationships between forest structure and the detection of flood inundation in forested wetlands using C-band SAR," *International Journal of Remote Sensing*, vol. 23, no. 3, pp. 443–460, 2002.

[18] L. Hess, J. Melack, and F. Davis, "Mapping of floodplain inundation with multi-frequency polarimetric SAR: Use of a tree-based model," in *Proceedings of International Geoscience and Remote Sensing Symposium*, vol. 2. IEEE International, 1994, pp. 1072–1073.

[19] L. Hess, J. Melack, S. Filoso, and Y. Wang, "Delineation of inundated area and vegetation along the Amazon floodplain with the SIR-C synthetic aperture radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 4, pp. 896–904, 1995.

[20] M. Horritt, "A statistical active contour model for SAR image segmentation," *Image and Vision Computing*, vol. 17, pp. 213–224, 1999.

[21] D. Mason and I. Davenport, "Accurate and efficient determination of the shoreline in ERS-1 SAR images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 34, no. 5, pp. 1243–1253, 1996.

[22] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models," in *Proceedings of First International Conference on Computer Vision*, London, 1987, pp. 259–269.

[23] A. Amini, S. Tehrani, and T. Weymounth, "Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints," in *Proceedings, Second International Conference on Computer Vision*, 1988, pp. 95–99.

[24] D. Williams and M. Shah, "A fast algorithm for active contours and curvature estimation," *CVGIP: Image Understanding*, vol. 55, no. 1, pp. 14–26, 1992.

[25] L. Cohen, "On active contour models and balloons," *Image Understanding*, vol. 53, pp. 211–218, 1991.

[26] C. Xu and J. Prince, "Gradient vector flow: a new external force for snakes," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Los Alamitos: Computer Society Press, 17–19 June 1997, pp. 66–71.

[27] ——, "Snakes, shapes and gradient vector flow," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 359–369, 1998.

[28] L. Ji and H. Yan, "Attractable snakes based on the greedy algorithm for contour extraction," *Pattern Recognition*, vol. 35, pp. 791–806, 2002.

[29] L. Cohen and I. Cohen, "Finite-element methods for active contour models and balloons for 2-D and 3-D images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 11, pp. 1131–1147, 1993.

[30] J. Ivins and J. Porrill, "Statistical snakes: active region models," *Image and Vision Computing*, vol. 13, no. 5, pp. 431–438, 1995.

[31] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd ed. Englewood Cliffs, New Jersey: Prentice Hall, 1988.

[32] NASA Landsat 7 Page. (2005, 15 Apr.) Comparison of Landsat Missions. [Online]. Available: http://landsat.gsfc.nasa.gov/project/Comparison.html

[33] J. Silva and H. Kux, "Remote sensing techniques to the detection and mapping of flooding dynamics within the Pantanal, Mato Grosso do Sul State, Brazil: preliminary results," in *Proceedings of the 24th international symposium on remote sensing of environment*, vol. 1, 27–31 May 1991, pp. 353–365.

[34] K. Sivaprasad and R. Bolus, "Delineation of 1993 midwest flooding using ERS-1 SAR, SPOT and Landsat imaginery," in *Proceedings of International Geoscience and Remote Sensing Symposium*, vol. 3. IEEE International, 1994, pp. 1439–1441.

[35] J.-H. Ryu, J.-S. Won, and K. D. Min, "Waterline extraction from Landsat TM data in a tidal flat, a case study in Gomso Bay, Korea," *Remote Sensing of Environment*, vol. 8, no. 3, pp. 442–456, 2002.

[36] S. Theodoridis and K. Koutroumbas, *Pattern Recognition.* Academic Press, 1998.

[37] ESRI, "ESRI shapefile technical description, an ESRI white paper," Environmental Systems Research Institute, 380 New York Street, Redlands, CA 92373-8100 USA, Tech. Rep., jul 1998.